ESC-TR-92-100

AD-A256 017

2

Technical Report
955

Neural Network Architectures for General Image Recognition

R.L. Harvey
P.N. DiCaprio
K.G. Heinemann

21 July 1992

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

Prepared for the Defense Advanced Research Projects Agency
under Air Force Contract F19628-90-C-0002.

This report is based on studies performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. The work was sponsored by the Defense Advanced Research Projects Agency under Air Force Contract F19628-90-C-0002.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

Hugh L. Southall

Hugh L. Southall, Lt. Col., USAF
Chief, ESC Lincoln Laboratory Project Office

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

NEURAL NETWORK ARCHITECTURES FOR GENERAL IMAGE RECOGNITION

R.L. HARVEY
P.N. DICAPRIO
K.G. HEINEMANN
Group 22

TECHNICAL REPORT 955

21 July 1992

LEXINGTON MASSACHUSETTS

ABSTRACT

As part of Lincoln Laboratory's research on neural network technology, a general-purpose machine vision system was designed that can learn to recognize diverse objects. This system models human vision, primarily with neural networks, and the learning is by example. The system was tested on two disparate classes of objects, military vehicles and human cells, with video images of natural scenes. These objects were chosen because large databases are available and because most researchers judge them to be unrelated. The system was trained and tested with 40 images of military vehicles. After training with 18 images, the system was able to recognize the tanks, howitzers, and armored personnel carriers of the remaining images without error. Pathologists at Lahey Clinic Medical Center collaborated in the cytology study where the system was trained and tested on 156 cell images from human cervical Pap smears. After training with 118 images, the system correctly classified all of the other cells as normal or abnormal (that is, precancer). These results are preliminary because the number of military vehicles and Pap smear samples was small. Nonetheless, the results are extremely encouraging and clearly indicate that additional development of the system is warranted. We note that the architecture of the system is applicable to many civilian and military tasks. The application of the system to a specific task requires appropriate training.

ACKNOWLEDGMENTS

The innovative research program at MIT Lincoln Laboratory supported this work from February 1989 to January 1990. Our supervisor, R.P. Rafuse, gave us much encouragement.

We wish to acknowledge our Lincoln Laboratory colleagues Mary Fouser, Pat Hirschler-Marchand, Paul Kolodzy, Gloria Liias, Courosh Mehanian, Murali Menon, and Alex Sonnenschein for their help and valuable discussions.

We wish to thank Alden Hayashi and Leslie Spiro for editing and improving the text during preparation of a Lincoln Laboratory Journal article. Many of their changes were incorporated in this report.

We are also grateful to Professor John Uhran Jr., of the University of Notre Dame, Dr. Barney Reiffen, and Professor Mike Carter of the University of New Hampshire for their consultation during this work. Special thanks to our Lahey Clinic Medical Center collaborators Drs. Mark Silverman and John Dugan.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS
(Continued)

LIST OF ILLUSTRATIONS
(Continued)

LIST OF ILLUSTRATIONS (Continued)

LIST OF ILLUSTRATIONS
(Continued)

LIST OF TABLES

1. INTRODUCTION

1.1 Motivation

Machine vision (MV) is a field in rapid development with ambitious goals. In its broadest definition, MV is the scientific discipline that addresses the problem of using sensor-derived images to recognize objects. An Innovative Research Project (IRP) project was designed to test some new ideas in MV. The main result of this project is a general architecture for MV, which is developed by modeling the human vision system primarily using neural networks (NNs).

To test the generality of the MV's architecture, images of military vehicles and Pap smear cells are employed. These images are chosen because, for most researchers, the two classes are quite different, and because ample size databases are available. The same vision system without significant changes processed the two classes of images. The results are encouraging. Tests show that after sufficient training, the system can recognize new images without error.

Good MV systems have many applications. Applications of interest to MIT Lincoln Laboratory are in remote sensing and automatic target recognition. Other uses include medical screening, industrial inspection, and robot vision. The vision architecture developed during this project is applicable to all these areas.

1.2 Conventional Machine Vision

To show the context of the work a brief review of conventional MV, an *active research* area for more than 30 years, is given. The section is from review articles. See Gordon, 1989 [1], Rosenfeld, 1987 [2], and Rosenfeld, 1988 [3] for more information.

To a major extent, conventional MV technology evolved from the work of one man, D. Marr, at MIT in the 1970s. Developed from the information theory, cybernetics, and digital computer technology of that era, Marr's contributions were made within the discipline of artificial intelligence. His work led to numerous papers on vision and, finally, to the book *Vision* [4].

An acknowledged contribution of Marr's was his attempt to clarify the thinking about vision systems or, more generally, information processing systems. In his work, Marr introduced the distinction among three levels of explanations: (1) the theory level, (2) the algorithm level, and (3) the implementation level. Consideration of these levels and to the issues raised by them leads to a sequence of questions that guides the design of an MV system.

Marr's explicit ideas somewhat puzzled researchers in vision at the time because other approaches used concepts that were undefined, or they used descriptions rather than explanations [1, pp. 194, 225]. Marr attained a high degree of rigor because his approach produced ideas that could be checked directly by computer simulation.

At the theory level, Marr asserted the key issue is determining both a goal for the computation and strategies for achieving that goal. By explicitly stating the computational goal and accompanying strategies, the machine's achievements can be described and its constraints characterized. Knowledge of the constraints, in turn, allows the processing stages to be defined.

At the algorithm level, the key issues are how the input and output are represented and the actual algorithm for transformation. The algorithm depends partially on the nature of the data representation.

At the implementation level, the key issue is how the machine actually works. The concern here is with the hardware of the machine and the nature and operation of its component parts.

For example, applying Marr's approach to optical sensors and two-dimensional processing leads to a modular design with the following consecutive processing stages:

1. Extract features such as edges from an image to produce a map representation. The map (called the primal sketch) consists of pixels and their feature values, such as edge strengths. (In this context, edge strengths are various combinations of first and second derivatives at each point in the image.)
2. Improve the map by grouping pixels in connected regions.
3. Represent the map by an abstract relational structure.
4. Recognize objects by comparing the structure with stored models.

Three-dimensional scenes are an extension of 2-D scenes. For the extension to 3-D, stages 1 and 2 should be replaced by a method to find the surface orientation of each pixel. The process will produce a representation map called the 2 1/2-D sketch.

Further extensions of Marr's method add one or more of the following stages:

1. Cleanup of input pixel values with image-restoration techniques.
2. Production of multiple images for stereomapping and motion analysis.
3. Adjustment of the processing by feedback from later stages to earlier stages.
4. Recognition of objects by matching them with models made up of composite parts.

The most rigorous of the conventional MV design methods comes from Marr's work. Yet, to some MV researchers and potential users, the performance of conventional MV systems is disappointing. The following quotes by Rosenfeld, a founder and a leading MV figure in a special issue of the IEEE Proceedings reviewing the field, and others reflect this assessment.

> "The prevalent opinion in machine vision today is that any significant increase in processing complexity for industrial applications must await the arrival of new special-purpose, parallel-type computer architectures [5, p. viii]."

"There is inadequacy in current theories [of machine vision]. There is paucity of well defined problems that have well defined rewards associated with them [5, pp. 189-90]."

"Another problem associated with current activities on the theory of machine vision is an underlying assumption that a theory of 'general' machine vision is achievable. This assumption may be false [5, pp. 189-90]."

"...standard vision techniques for feature detection, segmentation, recovery, etc., often do not perform very well when applied to natural scenes [3, p. 867]."

"Ideally, the [vision process] stages should be closely integrated; the results obtained at a given stage should provide feedback to modify the techniques used at previous stages. This is rarely done in existing vision systems, and as a result, little is known about how to design systems that incorporate feedback between stages [3, p. 868]."

"Little is known about visual knowledge representation or about flexible control structures for vision systems [3, p. 868]."

"...humans can recognize objects—even complex objects whose presence was unexpected—in a fraction of a second, which is enough time for only a few hundred 'cycles' of the neural 'hardware' in the human visual system. Computer vision systems have a long way to go before they will be able to match this performance [3, p. 868]."

In summary, current MV performance is significantly less than that of biological vision. Note the references to parallel-type architectures, feedback to earlier stages, and flexible control structures. Note also the need for well-defined problems, a general structure, and applications to natural scenes. The architecture developed during this project has these features.

1.3 Goals and Methods

The main goal of the project was to develop general MV architectures that would work on a variety of image types without significant changes in the algorithm. This robustness contrasts with the current practice of tailoring MV systems to specific applications. Such tailored systems have not performed well in situations unforeseen by the designers.

In many respects the human vision system, known for its high performance over a wide range of objects and situations, is far superior to current MV systems. Thus, in developing the architecture of the system, a decision was made to model selected functions of the human vision system. Although this idea was not new, the approach was: the entire system, module by module, was primarily modeled through the use of NNs. Features were incorporated that give the human vision system advantages over MV systems, namely, feedback, parallel architecture, and a flexible control structure. All individual modules and the major composite subsystems were tested.

Two new technologies made this study feasible: NN theory and a new class of computers. These technologies were not available in the 1970s when Marr was developing his method.

1.4 Report Road Map

Each section is independent. Section 2 summarizes the results. Section 3 reviews the information about biological vision systems used in modeling. Section 4 develops the architecture. Section 5 describes the software testbeds. Testbeds are developed for an IBM PC/AT, a Sun workstation, and a networked Sun-CONVEX combination. Section 6 gives the test results. Section 7 describes some extensions and applications. Section 8 gives the conclusions. The mathematical details are in the appendix.

2. SUMMARY OF RESULTS

2.1 Introduction

This section summarizes the results obtained with the testbeds during the project. Section 6 contains a more complete description and discussion. These results are preliminary because of the limited number of tested images. Nevertheless, the results are extremely encouraging and are clearly pointing toward further work on the system, especially on larger databases.

The system consists of a location channel and a classification channel working in parallel. To simplify the interpretation of results, the location and classification channels were tested separately.

For the classification channel, the objects were centered, or foveated, by hand so that the tests evaluated classifying under ideal conditions. Thus our classification results gave an upper bound on system performance because the location process introduces additional errors. Table 1 shows the classification results.

TABLE 1

A Summary of the Results

Classification Tests
Military Vehicles (56 Images)
Data: 42 × 42 × 8 Bit Video, 700-m Range
Train: 10 Tanks, 8 Howitzers
Test: 10 Tanks, 8 Howitzers, 4 APCs
Result: No Errors (Preliminary).
Pap Smear Cells (156 Images)
Data: 175 × 175 × 8 Bit Video, 400X Magnification
Train: 66 Normal, 52 Abnormal
Test: 26 Normal, 12 Abnormal
Result: No Errors (Preliminary)
Location Tests
Coarse
Binary Test Patterns All Found in Field-of-View after Training System on Representative Shapes
Pull-In
Single Cells: Pull-In ≤ 8 Steps
Multiple Cells: Pull-In ≤ 8 Steps in Most Cases

2.2 Military Vehicles

For military vehicle classification, the input images are 120 x 128 x 8 bit video images. The vehicles are at a range of 700 m. To characterize classification, the vehicle images were extracted by hand. The extracted images were 42 x 42 pixels. Figure 1 shows a tank image and an extracted box. The system was trained on images of 10 tanks and 8 howitzers. The system was tested on different images of 10 tanks, 8 howitzers, and 4 armored personnel carriers (APCs). The system classified all the test images without error.

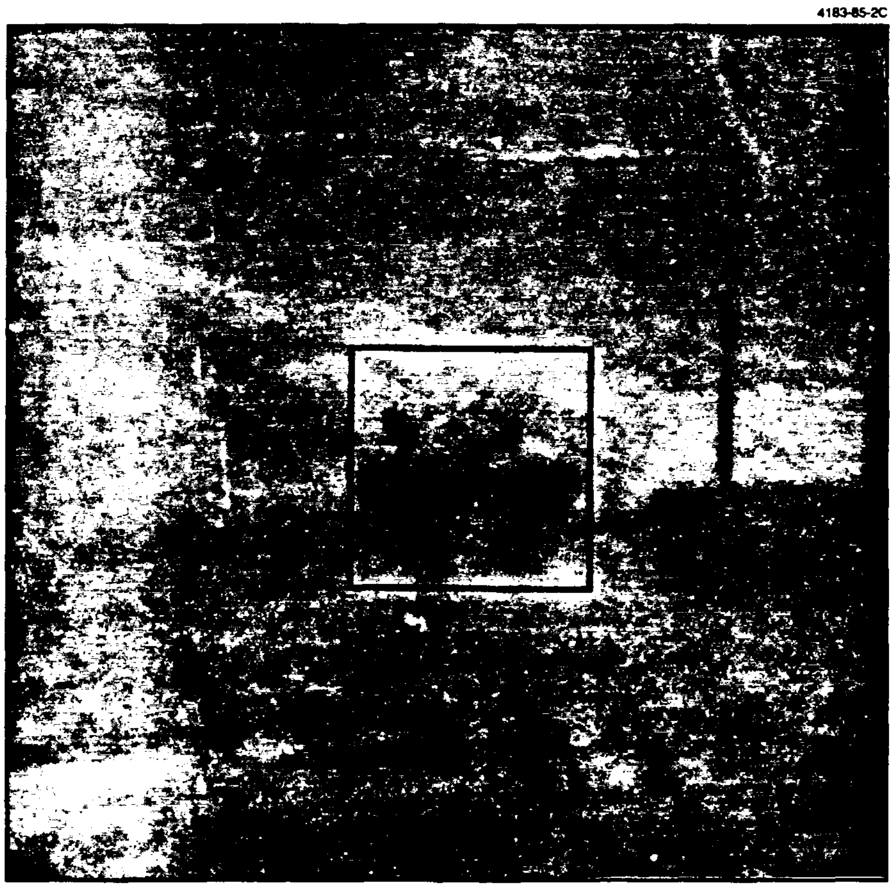


Figure 1. Example of a tank intensity image from a low-level TV camera and a 42 x 42 pixel box extracted for processing by the classification channel.

2.3 Pap Smear Cells

For the Pap smear cell classification, the input images are 512 x 512x 8 bit video images. The cells are at 400x magnification. Images of single cells were extracted again by hand. Each image was 175 x 175 pixels. Figure 2 shows a normal Pap smear cell. The system was trained on images of 66 normals and 52 abnormals. The system was tested on different images of 26 normals and 12 abnormals. It classified all these test images without error.

110566-2C

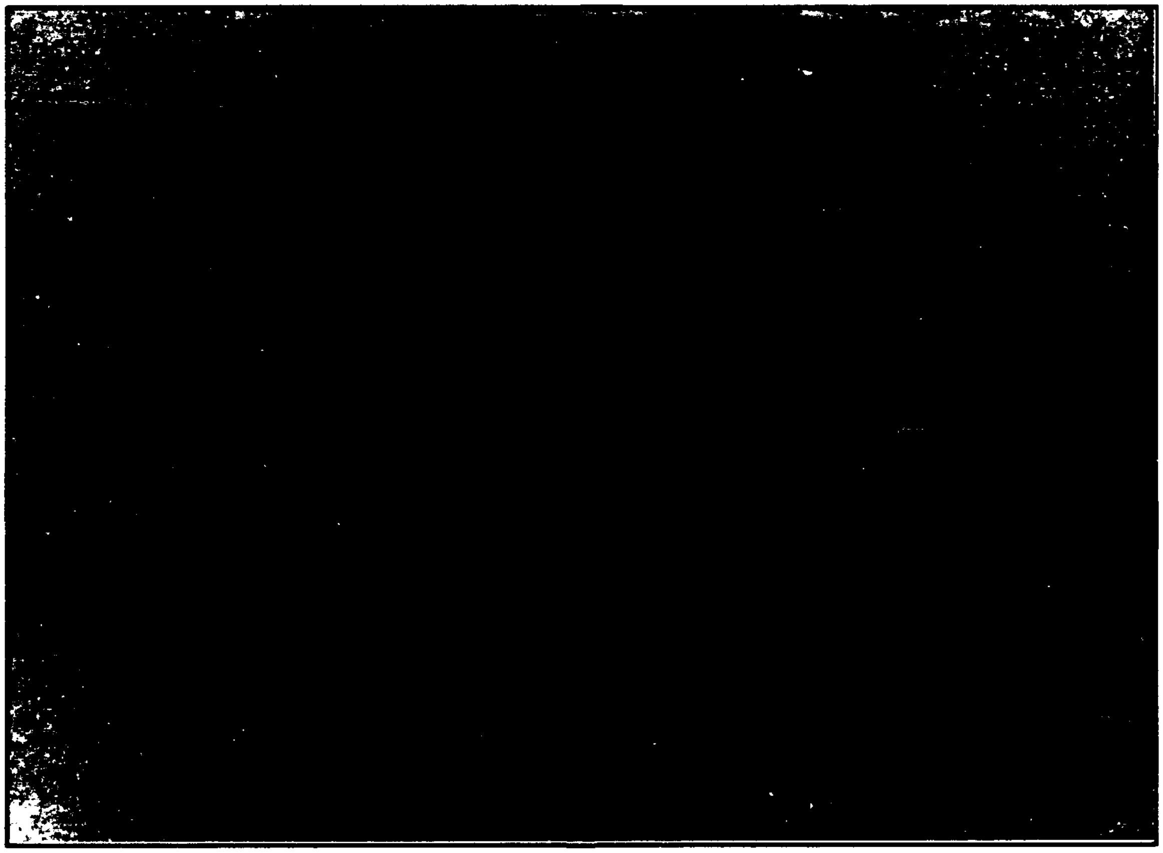


Figure 2. Typical image of normal Pap smear cells at 400x magnification. The cell size is about 175 x 175 pixels. Morphologic changes in the nucleus and cytoplasm, including size, shape, and texture, are criteria for malignancy.

For the location channel, the search function executes in two stages — coarse location followed by pull-in. In the coarse location, the system rapidly scans the field-of-view (FOV) for objects of interest. Typically, when found, an object is partially in the box. In pull-in, the system centers the object starting with the object partially in the box. Table 2 shows the location results.

For the coarse location, an IBM PC/AT test bed was used. The tests showed coarse location with 3 x 3 test patterns on a 25 x 25 pixel FOV. The results showed (1) coarse location of all test patterns when the system trained on representative shapes and (2) that the system is robust to offset and noise.

For pull-in, images of Pap smear cells in plain and natural backgrounds were used. For a single cell in natural backgrounds, pull-in was successful in eight or fewer steps, starting with a 175 x 175 pixel box about 40 percent off-center and ending within 10 percent off-center. For two cells situated side-by-side, a much more difficult case, pull-in was successful to one or the other cell in eight or fewer steps in most, but not all trials. Multiple cells were not exhaustively tested because of time limitations.

While these tests demonstrate the concept, more testing is clearly needed for identifying precisely where the system fails. Moreover, tests are needed for assessing applications with other types of images.

3. THE HUMAN VISION SYSTEM

3.1 Introduction

This project applied NNs and conventional processing to develop a general MV system based on a model of the human visual system. The idea is that with reasonable modeling, the performance would approach that of human beings.

Vision is an immensely active research area about which much has been written. This section summarizes selected aspects of human vision to provide background and to introduce terminology. In this report, vision means higher primate vision, especially human vision. Although data exists for several species, many of these results come from the macaque monkey, an animal with visual capabilities similar to that of human beings. In comparison, the vision systems of creatures lower than primates differ significantly.

3.2 General Brain Architecture

This discussion starts with an overview of the human brain [6-15]. The human brain has three main structures: the forebrain, midbrain, and hindbrain. The forebrain, or neocortex, shown in Figure 3 (top left), consists of two thin sheets of neurons (typically 1.5 to 5 mm thick) with well-known folds. Each sheet covers 1000 cm^2. (The midbrain and hindbrain are not shown.)

On the microscopic level, the neocortex consists of distinct areas or modules. The modules are defined by function and anatomical structure. A common designation scheme is the one by K. Brodmann (1909). Figure 3 (top right) shows some of the 52 Brodmann areas of the human brain by shading some of the areas associated with vision. More sophisticated experimental techniques indicate about 200 modules; the smaller subdivisions do not reflect real differences [7, p. 345].

3.3 Vision Architecture

The visual cortex occupies the entire back half of the neocortex hemisphere [15, p. 371]. Areas 17, 18, and 19 are feature detectors stages, areas 20 and 21 function as classifiers, and area 7b helps to locate objects in the FOV.

Figure 3 (bottom) shows the parts of the brain involved in vision. Certain modules in the midbrain belong to the visual system, for example, the lateral geniculate body or nucleus (LGN). Images captured by the eye balls are relayed through the LGN, which has processing functions and acts as a buffer.

The retina has about 1.25×10^8 receptors. Data compression by retinal processing is about 125 to 1, which gives a resolution near fovea, the most sensitive part of the retina, of about 1000×1000 pixels. The relative discrimination to brightness variations is 570 "just noticeable differences," and to frequency variations is 128 "just noticeable differences [14, p. 26]."

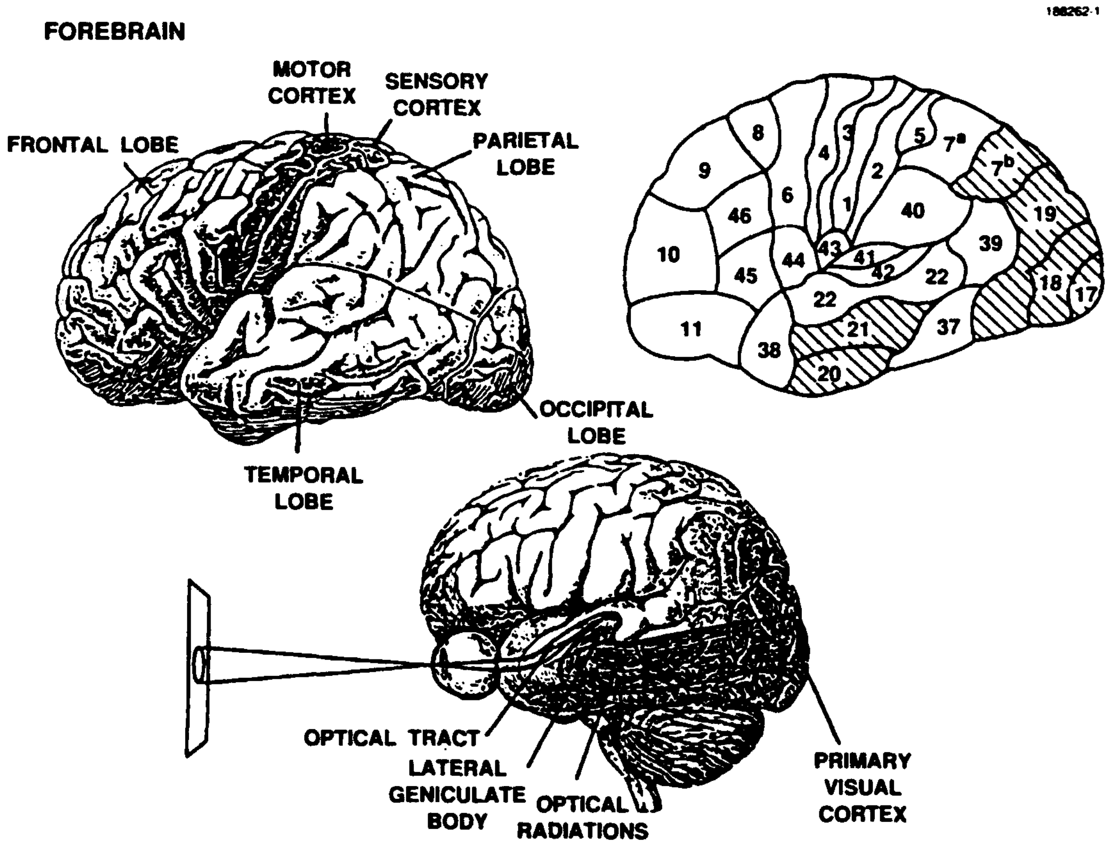


Figure 3. Schematics of the human brain showing the major structures and the Brodmann areas.

The FOV of each pixel at fovea with good contrast and brightness is about 0.5 min of arc (8 mdeg) [9, p. 45]. Thus, the FOV for highest acuity for each eye is 500 min of arc or 8.3 deg. The eyes can track with about 1 min of arc precision [9, p. 34].

Assuming 7 bits per pixel and a 100 Hz pulse frequency along the optic nerve (typical values), the data rate to the visual cortex is about 700 Mb/s, less than the capacity of current fiber optic channels.

Researchers have extensively studied the vision system of macaque monkeys [15, p. 37]. In the macaque and other primates, at least a dozen cortical areas are involved with vision. These areas are shown in Figure 4. Evidence shows a hierarchical structure for the vision system. That is, researchers can assign the modules to distinct processing levels. (Deductively, there is no reason to expect such organization. For example, the brain could be a complex network without distinct hierarchical levels [15, p. 371].)

Researchers have mapped over 30 pathways among vision-related areas, but the actual number is probably much larger because there are many connections to areas that have not been studied.

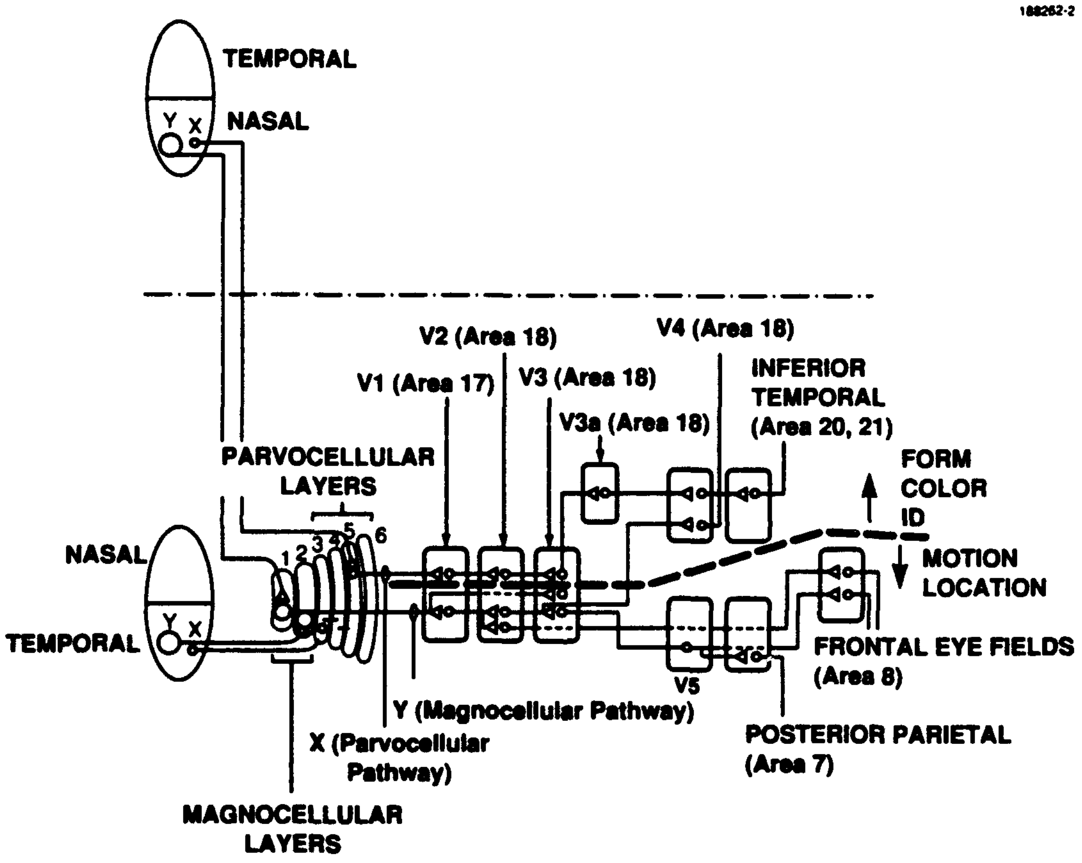


Figure 4. Schematic of visual projections to the brain areas involved in vision.

A basic finding is that with few exceptions the pathways connect the areas in reciprocal fashion [15, p. 371].

For the dozen visual areas, the overall cortical hierarchy has six levels. Researchers constructed the hierarchy by assigning each area to a level just above the highest area providing an input [15, p. 372]. The procedure leads to the six levels.

Anatomical, behavioral, and physiological data show two distinct channels for classifying and locating, shown in Figure 4 by the dotted line and the X-Y pathways. The two channels separate at the retina and remain separate at the cortical and midbrain levels. Evidence shows the classification channel analyzes form and color. The location channel analyzes visual motion across the FOV [15, p. 372].

The two channels have separate retinal detector cells, labeled X and Y. The X-cells go to the classification channel [6, p. 353]; they are medium sized cells with small optical fields, which give high-acuity and comparatively slow response times. The Y-cells go to the location channel; they are big cells with large optical fields, which give low acuity and fast response times. (Another kind of retinal cell, the W-cell, goes to the midbrain areas. The W- cells have small fields and coordinate the FOV to head and eye movements.) The population distribution is about X (50 percent), Y (5 percent), and W (45 percent).

The classification channel starts at the X-cells of the retina. It then goes to the RGC. It continues to the LGN, to the primary visual cortex (Vl — also called area 17), and to the secondary visual cortex (V2-V5 — also called areas 18 and 19). It then goes to the inferior temporal cortex (ITC — also called areas 20 and 21).

The visual pathway maps the FOV seen by the eyes onto Vl. The mapping in Vl is impressed on the fourth layer (out of six layers) of the cortex sheet. The mapping is continuous and has the well-known logarithmic distortion near fovea and rotates the external image about the horizontal axis.

Areas Vl, V2, and V3 work like feature detectors. As discovered by Hubel and Weisel (1955), the features provide only contour and orientation information about the pattern in the FOV.

Researchers measured the response of individual cells (neurons) along the classification channel to external images [10, p. 69]. In the retina and LGN, the response is either on-center/off-surround or off-center/on-surround. In Vl, three cell types have been found: simple, complex, and hypercomplex.

Simple cells respond to stationary edges, slits, or lines in the external image at precise orientation (angle). In human beings the angular resolution between two straight lines is about 10 deg [9, p. 48]. Interpolation allows discrimination between two lines that differ by about 3 deg. To produce a response from a simple cell, a stationary line must be carefully oriented and positioned.

The retinal cells on which light must fall to affect a neuron in V1 are clustered in a small area called the receptive field. At fovea, simple cells have receptive fields measuring about one-quarter deg by one-quarter deg. At the periphery, the receptive field size is about one deg by one deg.

Complex cells respond to moving edges, slits, or lines in a specific direction. About 75 percent of cells in Vl are complex [10, p. 74]. The receptive field of complex cells is slightly larger than that for simple cells. Near fovea the receptive field size is about one-half deg by one-half deg.

Hypercomplex cells respond if one or both ends of a line stop in the receptive area. If the line goes through the receptive area without stopping, the hypercomplex cell response goes to zero or a constant value.

Summarizing the feature detectors, the features in primate vision are stationary or moving edges, slots, or lines. In comparison, these are not the features commonly used in current machine vision algorithms. Typical features in MV algorithms are corners, faces, frequencies, or responses of matched filters.

The ITC (areas 20 and 21) in the classification channel works like a classifier. The ITC output goes to higher level centers of logic and emotion [16, ch. 5].

The location channel starts at the Y-cells of the retina. The channel goes through the midbrain areas to the superior colliculus (SC) and then to the pulvinar nucleus (PT). It continues to the posterior parietal (PP). The output of this channel goes to the frontal eye fields (area 8). Evidence suggests this system is responsible for location analyses of objects in the FOV.

Besides connections along the *classification* and location channels, the two channels are interconnected. The SC projects directly to the ITC bypassing the rest of the main route. The PT projects to the secondary visual cortices and to the ITC. The classification channel also passes data to the location channel bypaths from Vl to SC and PT [16, p. 101].

Considering the system as a whole, researchers believe it works as follows: retinal images send information for pattern analysis by the classification channel. The classifying system passes the information from the RGC to the LGN to Vl, through several paths in V2, V3, V4, and V5, then to the ITC.

Simultaneously, retinal images pass through midbrain routes, which locates objects and analyzes spatial relations and motion. The location system interacts with the pattern analysis system at most steps along the route. Major interactions for constructing spatial relations occur in the location visual cortex [16, p. 101].

Besides the preceding step-by-step level scheme, there also is a relationship between the level and the receptive-field size. Receptive fields are smallest in Vl, and they increase in size at successive levels of the hierarchy [15, p. 372].

Researchers have found that receptive field properties can vary [16, ch.5]. That is, there is a mechanism for adjusting a feature extracting cell. The mechanism employed probably is presynaptic inhibition. Presynaptic inhibition is a mechanism that acts selectively to turn off particular inputs to a neuron. (For a description of synaptic processes see [12, chs. 9 and 10].)

Presynaptic inhibition in the feature extraction stages changes the shape and location of the receptive field. Research shows emotional states can alter the receptive fields. Under normal conditions, stimulating the ITC changes the receptive fields of area Vl, suggesting that the ITC may exert feedback control over the feature detectors.

Thus, recognition is likely to involve an active feedback process that restructures the feature extraction stage. Feedforward and feedback signals continue until matching occurs between an input and some known class of stimulus [16, p. 108].

Moreover, recognition starts with standard receptive fields. If matching by the ITC fails, feedback then shifts the processes in the preceding stages to extract features for another object class.

Research also suggests a mechanism for directing attention to selected locations in the FOV. In short, there is windowing. Windowing focuses on small details and also ends notice of other objects in the FOV. Researchers suspect the midbrain directs this process, perhaps cued by cortical inputs.

The vision system has other inputs besides those from the retina. Signals from the motor systems provide feedback about eye position. Inputs from the frontal cortex may be the source of selective attention. These attention inputs direct goal-related processes. Visual memory is another source of input, which may improve the search strategy in the perceptual analysis [16, p. 132].

Research shows extensive use of feedforward signals. Outputs of the visual front-end feed a variety of later stages. The higher processing levels may be tapping the preprocessor signals for simple information, like overall brightness.

Output is available from each stage of processing probably to all parts of the system. The primary cortex provides information about the detailed nature of the visual field and its spatial structure, but it does not analyze patterns into objects. To organize patterns, the higher levels draw on experience (memory).

After visual processing, the outputs from the visual system are used throughout the brain. One area records visual objects (memory). Another area organizes a logical world model from experience and sensory inputs (cognition). Yet another area responds emotionally to objects perceived (motivation) [16, ch. 5].

Summarizing the above description, Figure 5 shows a block diagram of the brain's visual processing. As seen, the system is a sensor-preprocessor-feature-extracting classifier system. A small number of serial stages process large arrays of data. Within each stage the processing is heavily parallel. The modules are connected by feedforward and feedback pathways.

Finally, the system is asynchronous, that is, there is no master clock. The system is a serial-parallel, analog, asynchronous, real-time computing machine while man-made computers are usually serial, digital, synchronous, and off-line. The system continually updates on each pathway. Thus, there will be differences in the processing time among the modules. In practice the longest times are a fraction of a second, roughly, the characteristic visual interaction time with the outside world.

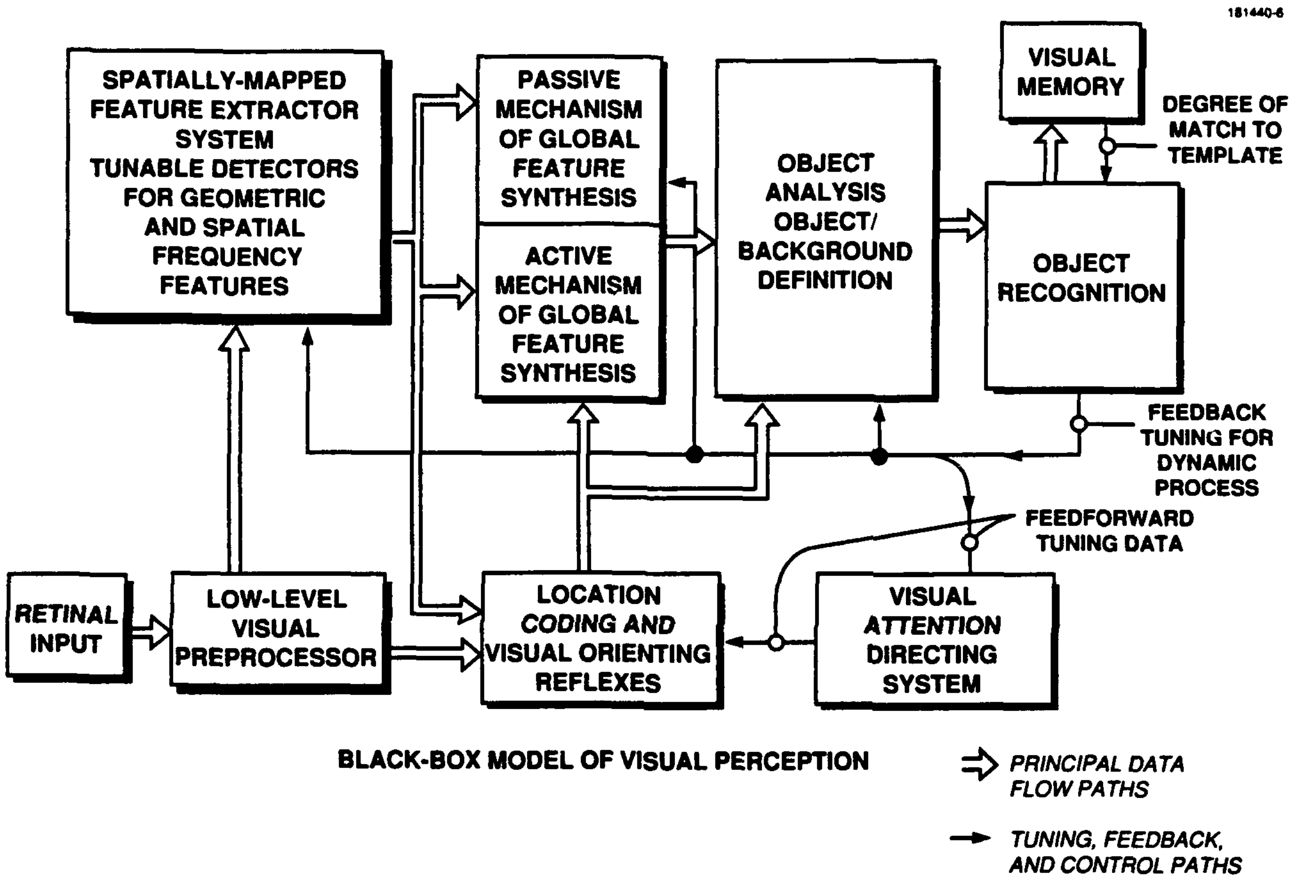

181440-6
SPATIALLY-MAPPED FEATURE EXTRACTOR SYSTEM TUNABLE DETECTORS FOR GEOMETRIC AND SPATIAL FREQUENCY FEATURES
PASSIVE MECHANISM OF GLOBAL FEATURE SYNTHESIS
ACTIVE MECHANISM OF GLOBAL FEATURE SYNTHESIS
OBJECT ANALYSIS OBJECT/ BACKGROUND DEFINITION
VISUAL MEMORY
DEGREE OF MATCH TO TEMPLATE
OBJECT RECOGNITION
FEEDBACK TUNING FOR DYNAMIC PROCESS
FEEDFORWARD TUNING DATA
RETINAL INPUT
LOW-LEVEL VISUAL PREPROCESSOR
LOCATION CODING AND VISUAL ORIENTING REFLEXES
VISUAL ATTENTION DIRECTING SYSTEM
BLACK-BOX MODEL OF VISUAL PERCEPTION
PRINCIPAL DATA FLOW PATHS
TUNING, FEEDBACK, AND CONTROL PATHS

Figure 5. A block diagram of the vision system containing modules discussed in the text [Redrawn from Kent (1981)].

4. THE APPROACH — A BIOLOGICALLY INSPIRED TWO CHANNEL SYSTEM

4.1 Introduction

Section 1 discusses the advantages of biological vision over current MV. Section 3 suggests these advantages are attributable to feedback, flexible control, and the kinds of feature detectors. This section describes our architecture for a general purpose MV system, which incorporates some of the biological characteristics.

The purpose of the system is to find and identify spatial patterns of luminance in the FOV. The term "general purpose" means recognizing objects from widely varying classes without changing the algorithm. In practice, applying the system to a class requires setting a few sensitivity parameters and then training the system by examples.

Our approach was to model the human vision system. Thus the functions and names of different modules were based on structures in the human brain, described in Section 3. The modules roughly approximate the functions of their biological counterparts, as understood at this time. For convenience, a mixture of NNs and standard processing algorithms were used to implement the module functions.

All known characteristics of human vision were not modeled. As shown in Table 2, modeling of binocularity, size invariance, motion detection, color sensitivity, and virtual boundary perception were omitted so that a simple architecture could be studied. Moreover, these features are unnecessary for many applications. In principle, modules that model these characteristics can be added to the testbed system (see Section 7). The architecture of the system includes two major channels that work together. The location channel searches for objects of interest in the FOV and, after one is found, the classification channel classifies it. Studies of the human vision system, as well as that of other animals, suggest that the locating and classifying functions are separate (see Section 3).

Figure 6 shows a block diagram of the architecture. Sections 4.2 and 4.3 describe each of the modules. To illustrate how the system works, a 525 x 525 pixel input image with 8 bit pixels is used. It is assumed that 175 x 175 pixel objects appear in this image. This object size corresponds to normal cells in a Pap smear image at 400x magnification. In operation, the location and classification channels are coupled and work together simultaneously; however, for convenience, the description starts with the classification channel and the modules shared by the two channels. The text follows the overall block diagram (Figure 6).

4.2 Classification Channel

Certain classification channel modules approximate the functioning of selected brain areas: the lateral geniculate nucleus (LGN), visual area 1 (V1, also called A17), visual area 2 (V2, also called A18), inferior temporal cortex 1 (ITC1, also called A20), and inferior temporal cortex 2

TABLE 2

Assumptions for Modeling the Human Vision System

Input Images
Up to 525 × 525 Pixels Gray (8 Bits)
Architecture
Include Principle Data Flow Paths Feedback and Control Paths Two Channel Architecture Neglect Color Motions Binocularity Retina Processing Perceived Boundaries

(ITC2, also called A21). Other modules, such as the SUM module, approximate certain biological functions without the anatomical correspondence.

4.2.1 LGN Module-Grayness Processing

Figure 7 shows the front-end processing stages in the classification channel. The classification channel has feedforward and feedback signals. As shown, signals flow from the input image, through the feature extracting stages, and to ITC1 input.

The first module in the classification channel is the LGN, containing the CALIBRATE and NORMALIZATION boxes (Figure 6). Assume the location channel has found an object in the FOV (see Section 4.3). The location channel sends the pattern's position to the LGN module (Figure 7, bottom). In the example, the windowed image covers 175 x 175 pixels.

The CALIBRATE box computes a histogram of the windowed image. Our testbed images have 8-bit pixels. Histograms of these images are typically concentrated in a small band within the range of 0 to 255. Calibrating spreads the intensity values over the entire 0 to 255 range by linearly mapping the lower part to 0 and the upper part to 255.

The histogram's lower limit is defined as the point where the cumulative count is 1 percent of the peak value, and the upper limit is where the cumulative count exceeds 99.25 percent. These limits prevent outlying pixel values from affecting the histogram stretching factors.

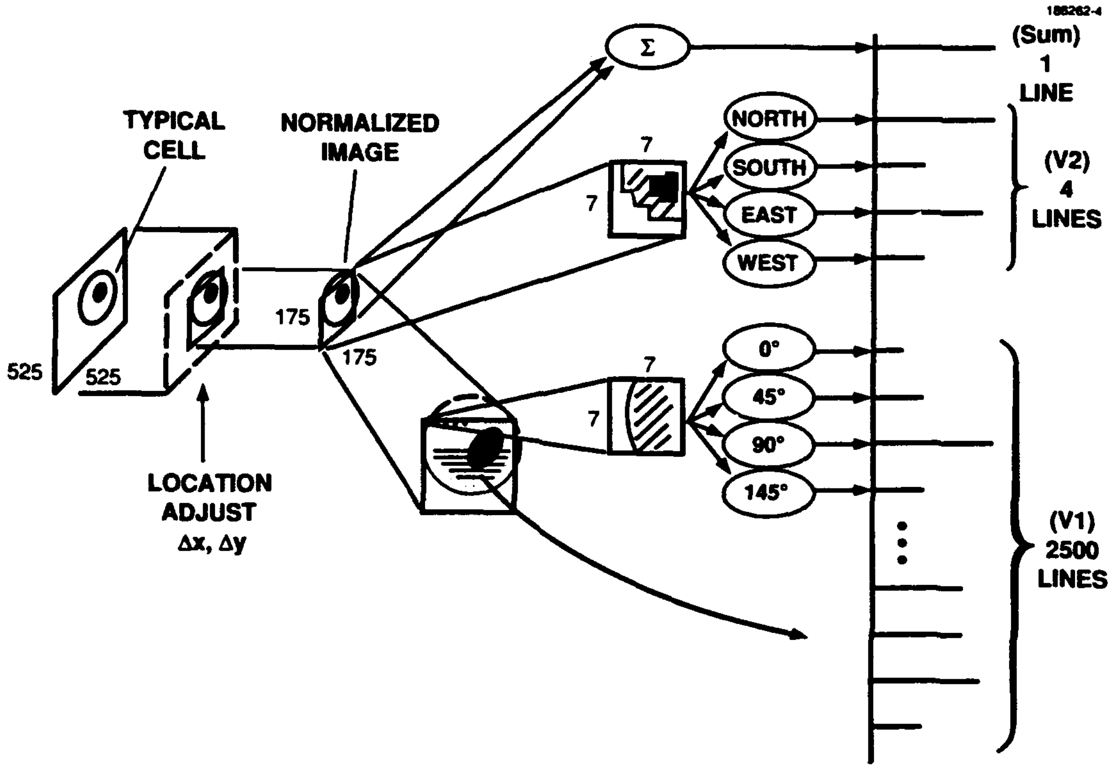


Figure 7. Summary of image processing operations. For the example of a 525 × 525 input image, the classification channel places a window of 175 × 175 pixels around the object in the image. (The window size is set to fit the object size.) The 175 × 175 window is then broken into subwindows to extract details of the image, and the details are stored in the feature vector.

The NORMALIZE box rescales the pixel values so pixels leaving the LGN module are in the range from zero to one by dividing the calibrated pixel values by 255.

The CALIBRATE and NORMALIZATION boxes handle the grayness and adjust for overall brightness in the FOV, which decouples the image's grayness and illumination from the rest of the processing. The decoupling allows the remaining modules to be designed for pixel values lying in a well-defined range, in our case from zero to one.

4.2.2 V1 Module-High Resolution Features

The first feature-generating module is A17 (or V1), which breaks the input window into subwindows of 7 x 7 pixels. Thus, in our example of a 175 x 175 pixel window, there are 625 subwindows. Note that the 7 x 7 subwindow size is unrelated to the input image size. Each 7 x 7 subwindow is then processed by SPIRAL MAP and VISAREA1.

SPIRAL MAP (Figure 6) scans through the subwindows in a spiral pattern. The mapping proceeds as follows: left to right across the top row, down the right column, right to left across the bottom row, up the left column, back across the second row, and so forth until the process ends at the center subwindow. The purpose of the spiral mapping is to simplify interpretation of the feature data.

VISAREA1 (Figure 6) does the high-resolution feature extraction. For each 7 x 7 pixel subwindow, VISAREA1 measures luminance gradients (increasing or decreasing) in four directions. A gradient is a characteristic of gray images and is analogous to an edge in a binary image. The luminance gradient in our system is the rate of change, or slope, in brightness across a 7 x 7 subwindow. Windows with an abrupt step in brightness in one direction will have a large gradient in that direction; windows with a gradual change in brightness from one side to the other will have a small gradient; windows with uniform brightness, that is, windows with no visible edges, will have zero gradient.

The gradients in different directions are usually not the same because the slope depends on direction. For each 7 x 7 subwindow, the system produces gradients in four directions: vertical, horizontal, and the 45-deg diagonals.

Figure 8 shows the operations for producing the four features of each 7 x 7 subwindow. As shown, with reflecting and rotating the input, only two different NNs are needed.

The gradient detectors in the system use a cooperative-competitive NN. In the test bed, these NNs have 25 hidden neurons and 1 output neuron. As suggested by biology [7], each neuron is either excitatory or inhibitory, not both. Figure 9 illustrates this NN.

The gradient-measuring NNs are designed to give responses to selected patterns. Figure 10 shows the design patterns. For the testbed, each feature detector NN has 1924 fixed interconnecting weights.

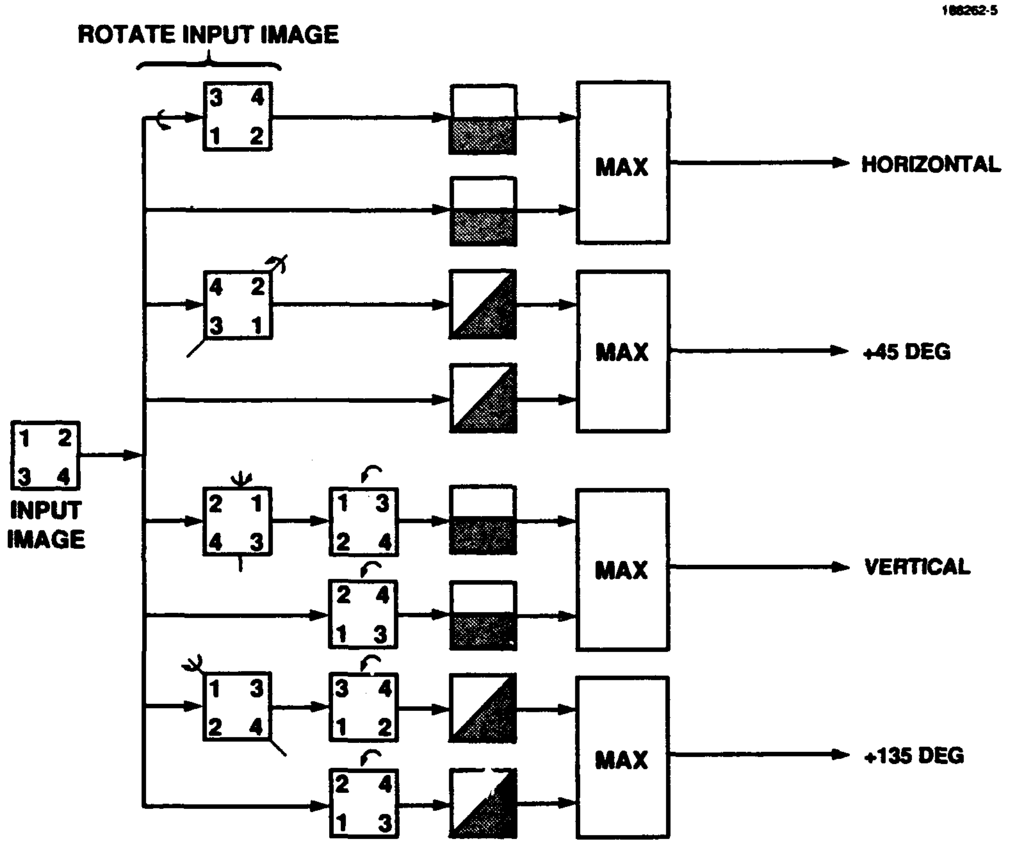


Figure 8. Feature detector module for the V1 module. The gray input image is reflected and rotated as shown so that only two neural networks are needed for the four directions and their two gradients.

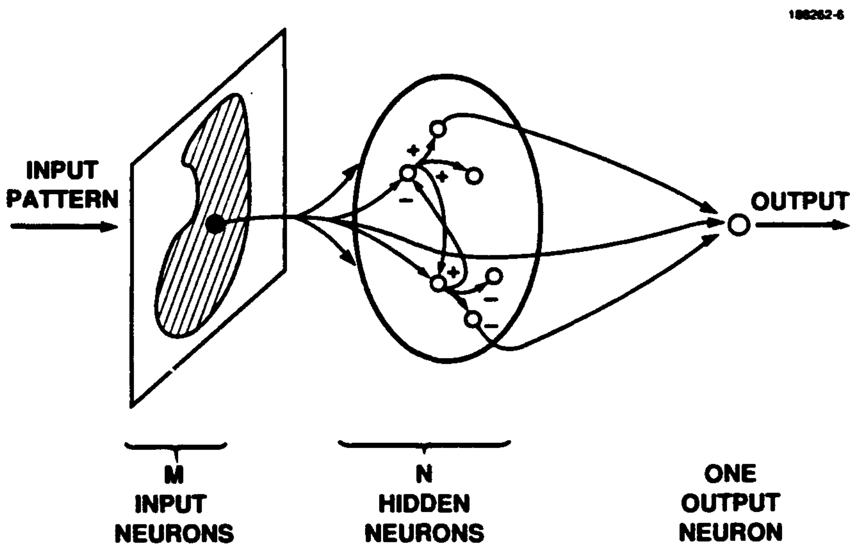


Figure 9. Architecture for a feature detector neural network. An input pattern, shown in cross-hatching, is impressed on M neurons. Each input neuron is connected to N hidden neurons and a single output neuron, whose output is high for a chosen angular orientation of the input pattern and low for other orientations. The hidden neurons may be excitatory (labeled +) or inhibitory (labeled −) and are interconnected. In the baseline system the input pattern is 7 × 7 pixels with 25 hidden neurons.

The weights are computed off-line with a genetic algorithm technique described in the appendix. The weights are fixed. Figure 11 shows the horizontal responses to the design patterns and Figure 12 shows the diagonal responses.

The gradient detectors model similar biological processes (see Section 3.3). Classification uses only these features, and those of V2 and SUM (see below). In the example, after windowing to 175 × 175 pixels, four orientation signals are produced for each 7 × 7 pixel subwindow giving 2500 feature lines from V1.

Note the system does not use operations common to other systems. It uses no Fourier transform. Nor does it have face, corner, circle detectors, or matched-filters.

To help interpret the feature values obtained from SPIRALMAP and VISAREA1, the V1 outputs are arranged in a vertical vector (Figure 7). For the 175 × 175 pixel window, there are 2500 V1 values because each 7 × 7 subwindow produces four values. Feature values are stored from the image's outer parts at the top of the vector and feature values from inner parts at the bottom of the same vector. Thus, data about the general shape of an image are found at the top of the vector and data about the interior are at the bottom.

181440-8

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
.5	.5	.5	.5	.5	.5	.5
.5	.5	.5	.5	.5	.5	.5
1	1	1	1	1	1	1
1	1	1	1	1	1	1

1

0	0	0	0	0	0	0
0	0	0	0	0	0	0
.25	.25	.25	.25	.25	.25	.25
.5	.5	.5	.5	.5	.5	.5
.75	.75	.75	.75	.75	.75	.75
1	1	1	1	1	1	1
1	1	1	1	1	1	1

2

1	1	1	1	1	1	1
.75	.75	.75	.75	.75	.75	.75
.5	.5	.5	.5	.5	.5	.5
.25	.25	.25	.25	.25	.25	.25
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

3

0	0	0	0	0	0	0
0	0	0	0	0	0	1
0	0	0	0	0	1	1
0	0	0	0	1	1	1
0	0	0	1	1	1	1
0	0	1	1	1	1	1
0	1	1	1	1	1	1

4

0	0	0	0	0	.25	.25
0	0	0	0	.25	.25	.75
0	0	0	.25	.25	.75	.75
0	0	.25	.25	.75	.75	1
0	.25	.25	.75	.75	1	1
.25	.25	.75	.75	1	1	1
.25	.75	.75	1	1	1	1

5

1	1	1	1	1	1	0
1	1	1	1	1	0	0
1	1	1	1	0	0	0
1	1	1	0	0	0	0
1	1	0	0	0	0	0
1	0	0	0	0	0	0
0	0	0	0	0	0	0

6

0	0	.25	.5	.75	1	1
0	0	.25	.5	.75	1	1
0	0	.25	.5	.75	1	1
0	0	.25	.5	.75	1	1
0	0	.25	.5	.75	1	1
0	0	.25	.5	.75	1	1
0	0	.25	.5	.75	1	1

7

1	1	.5	.5	0	0	0
1	1	.5	.5	0	0	0
1	1	.5	.5	0	0	0
1	1	.5	.5	0	0	0
1	1	.5	.5	0	0	0
1	1	.5	.5	0	0	0
1	1	.5	.5	0	0	0

8

1	1	.75	.5	.25	0	0
1	1	.75	.5	.25	0	0
1	1	.75	.5	.25	0	0
1	1	.75	.5	.25	0	0
1	1	.75	.5	.25	0	0
1	1	.75	.5	.25	0	0
1	1	.75	.5	.25	0	0

9

0	0	0	0	0	0	0
1	0	0	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	0	0
1	1	1	1	0	0	0
1	1	1	1	1	0	0
1	1	1	1	1	1	0

10

.5	.5	0	0	0	0	0
.5	.5	.5	0	0	0	0
.5	.5	.5	.5	0	0	0
1	.5	.5	.5	.5	0	0
1	1	.5	.5	.5	.5	0
1	1	1	.5	.5	.5	.5
1	1	1	1	.5	.5	.5

11

.25	.25	0	0	0	0	0
.75	.25	.25	0	0	0	0
.75	.75	.25	.25	0	0	0
1	.75	.75	.25	.25	0	0
1	1	.75	.75	.25	.25	0
1	1	1	.75	.75	.25	.25
1	1	1	1	.75	.75	.25

12

Figure 10. Gray patterns used to design the V1 feature detectors, as described in the appendix. The set includes horizontal (1-3), 45-deg diagonal (4-6), vertical (7-9), and 135-deg diagonal (10-12) with two gradient directions.

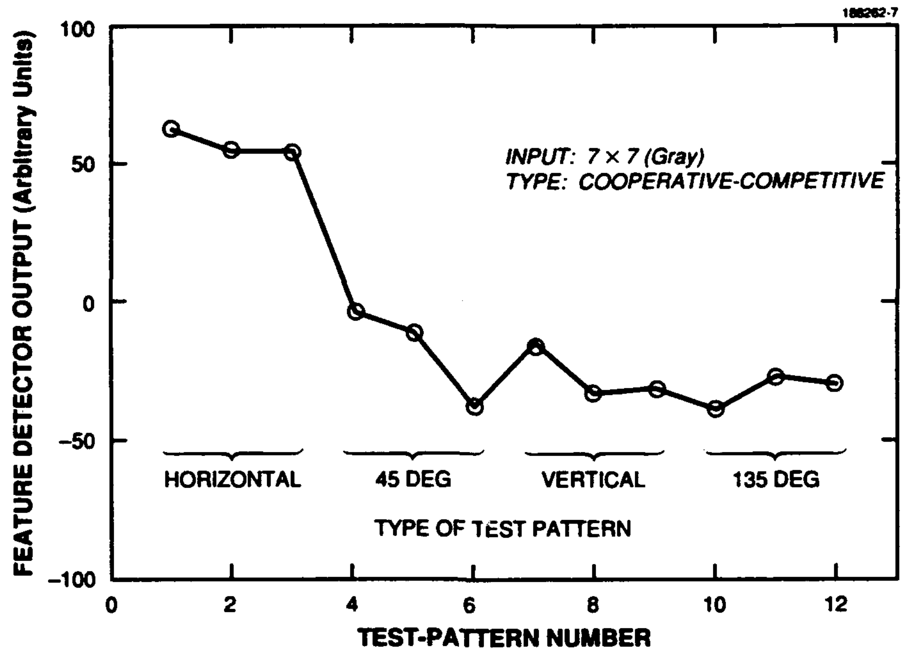


Figure 11. Response of the horizontal feature detector to the design patterns.

4.2.3 V2 Module-Shape Features

The second feature-generating module is A18 (or V2), which detects edges near the perimeter of the input window. V2 (Figure 6) is also part of the location channel (see Section 4.3), and its output contains information about an object's general shape.

To detect edges, the AVERAGE box (Figure 6) defocuses the image to produce a single 7 x 7 image, regardless of the size of the input image. For the 175 x 175 input case, the defocusing averages over 25 x 25 input pixels to produce each output pixel. The averaging smears pattern details (details captured by V1), but retains data about the outside edges.

Figure 13 illustrates averaging a cell wholly in the window (top) and partially in the window (bottom). As shown, the averaging produces a single smeared 7 x 7 pixel image of the pattern in the window.

VISAREA2 detects edges near the four sides of the defocused image. In VISAREA2, a 3 x 7 pixel detector detects the presence of near-horizontal edges at the top and bottom of the smeared image. A 7 x 3 pixel detector detects the presence of near-vertical edges at the right and left of the smeared image. Figure 14 shows the positioning of the detectors.

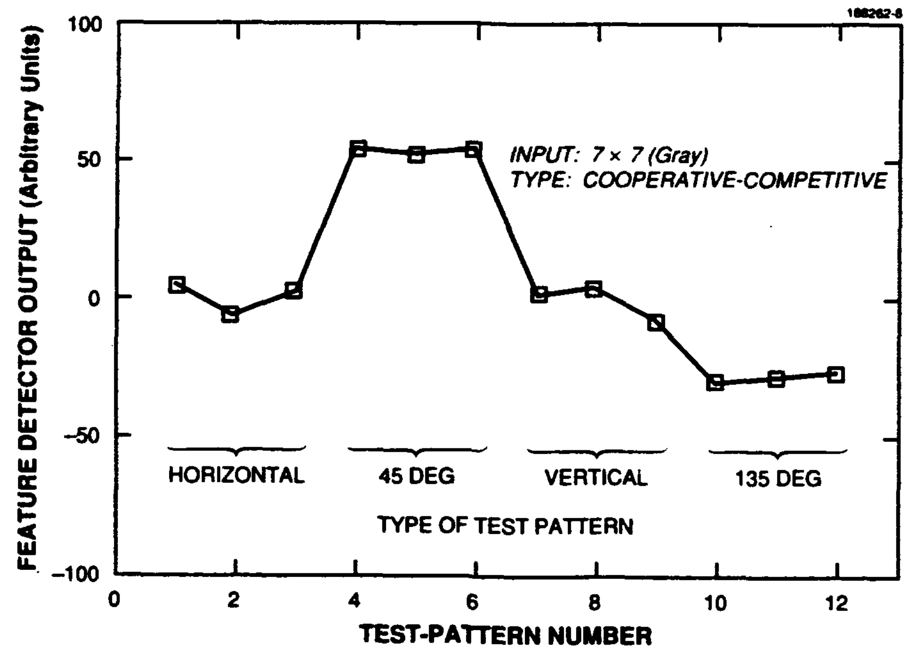


Figure 12. Response of the 45-deg diagonal feature detector to the design patterns.

For objects framed by the window, there will be edges on the four sides. The VISAREA2 output is four values. The values measure the NORTH, SOUTH, EAST, and WEST edge strengths. These four values are included in the feature vector. As shown in Figure 15, a single 7 x 3 NN, with rotations and complementing, can do all V2 feature detection.

Figure 16 shows the design patterns for VISAREA2. These patterns were used to design a 7 x 3 pixel input NN with the genetic algorithm technique described in the appendix. Patterns 1 to 4 are representative edges of a gray image. Patterns 5 to 8 are representative of nonedges, that is, parts of patterns that are not properly windowed.

The VISAREA2 edge detectors also use 25 neuron, cooperative-competitive NNs with fixed weights. Figure 17 shows the response of the V2 vertical (WEST) edge detector module to the design patterns. The presence of an edge in this section of the image (patterns 1-4) gives a large response, while the others (patterns 5-8) give a much smaller response.

4.2.4 SUM Module-Object Size

The third feature-generating module is SUM (Figure 6), which adds up the pixel values of the input window. Thus, the single output from SUM measures the object's gross size. For convenience,

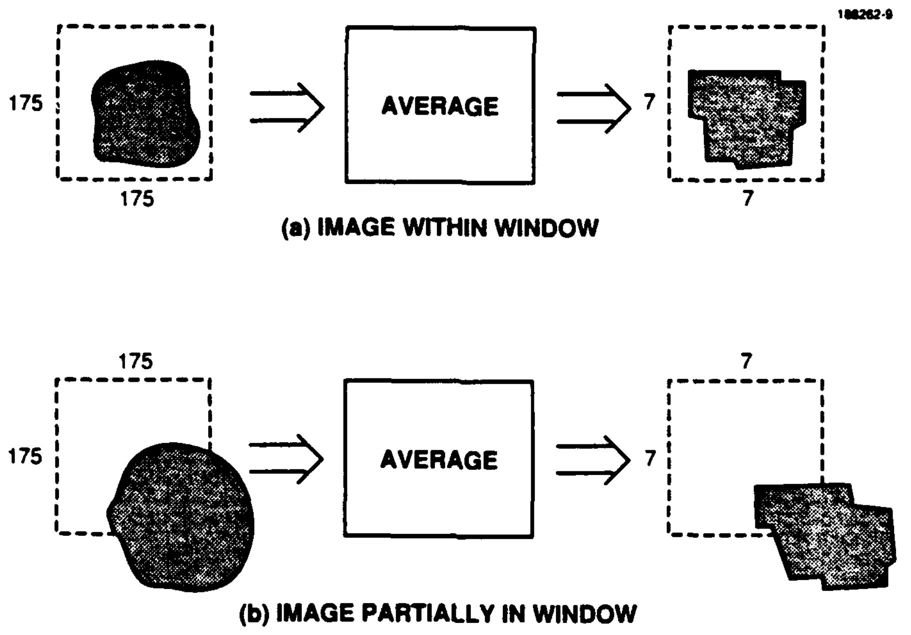


Figure 13. The AVERAGE module in V2 takes a, say, 175 x 175 pixel input image and smears its details to give a 7 x 7 pixel output image that retains the general shape information.

this function has been separated from V1 and V2, although the biological function occurs in both V1 and V2 (see Section 3.3). The SUM output also is included in the feature vector.

4.2.5 Feature Vector

The system classifies using data about detailed structure (V1), overall shape (V2), and size (SUM). The different subwindow sizes of V1, V2, and SUM model approximately the different size receptive areas of the visual cortex. For the 175 x 175 pixel window, there are 2500 values from V1, four values from V2, and one value from SUM. These 2505 values form the feature vector.

The SUM, V1, and V2 values are adjusted to give, roughly, equal influence to an object's size, shape, and detail structure, as described in Section 6.

Figure 18 shows the feature vector of a normal Pap smear cell. Note the relative sizes of the SUM, V2, and V1 components. Figure 19 shows V1 features near the center of a normal cell and an abnormal cell. The grid, not to scale, suggests the matrix of 7 x 7 subwindows. Note the differences between feature vectors for the two types of cells. The comparison suggests that an adequately sensitive classifier can distinguish the two cells — a suggestion that must be verified by testing.

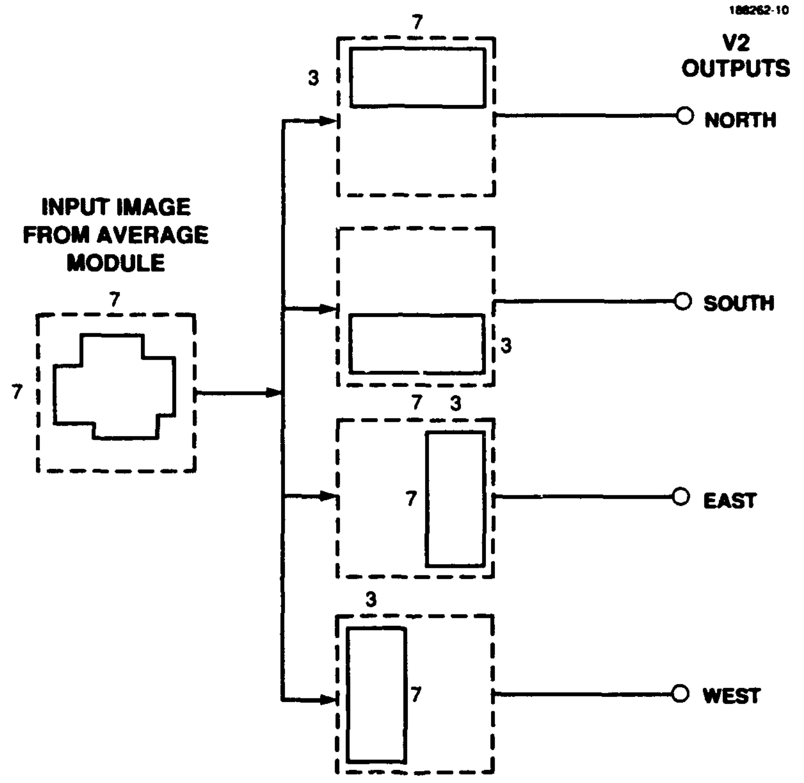


Figure 14. Block diagram of the V2 VISAREA2 modules. The modules detect the edges (if any) in 3 × 7 and 7 × 3 pixel sections of the 7 × 7 input image. The four outputs indicate the presence or absence of an edge in the corresponding section of the 7 × 7 input image.

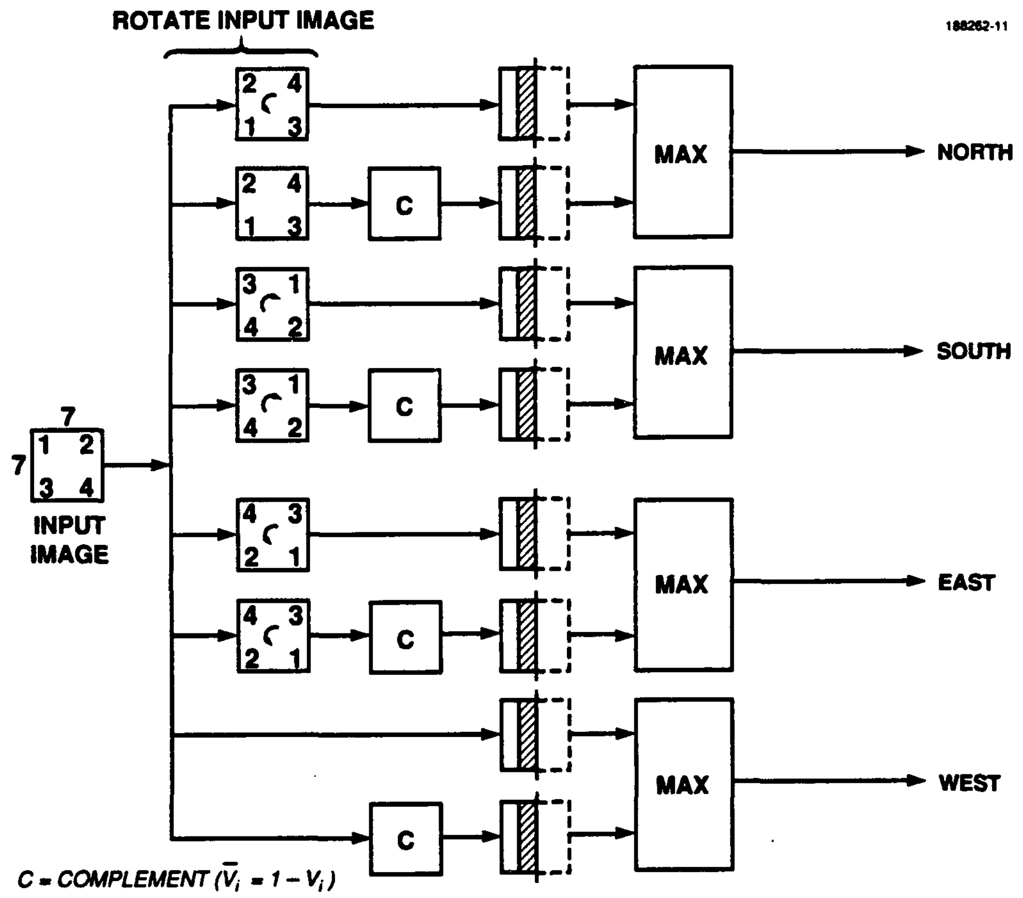


Figure 15. Details of the V2 VISAREA2 modules. The gray input image is rotated and complemented as shown so that only one NN is needed for detecting edges in the four sections of the image.

181440-7

$$\underset{1}{\boxed{\begin{matrix}0&0&0\\0&0&0\\0&0&.5\\0&.5&1\\0&.5&1\\0&0&.5\\0&0&0\end{matrix}}}\quad\underset{2}{\boxed{\begin{matrix}0&0&0\\0&0&.5\\0&.5&1\\0&.5&1\\0&.5&1\\0&0&.5\\0&0&0\end{matrix}}}\quad\underset{3}{\boxed{\begin{matrix}0&0&0\\0&0&.5\\0&.5&1\\.5&1&1\\.5&1&1\\0&.5&1\\0&0&0\end{matrix}}}\quad\underset{4}{\boxed{\begin{matrix}0&0&0\\0&0&0\\0&0&.5\\0&.5&1\\0&.5&1\\0&.5&1\\0&0&.5\end{matrix}}}$$

$$\underset{5}{\boxed{\begin{matrix}0&0&0\\0&0&0\\0&.5&.5\\0&.5&1\\0&.5&1\\0&.5&.5\\0&0&0\end{matrix}}}\quad\underset{6}{\boxed{\begin{matrix}0&0&0\\0&0&0\\0&0&0\\0&0&0\\0&.5&.5\\0&.5&1\\0&.5&1\end{matrix}}}\quad\underset{7}{\boxed{\begin{matrix}0&.5&.5\\0&.5&1\\0&.5&1\\0&.5&1\\0&.5&1\\0&.5&1\\0&.5&.5\end{matrix}}}\quad\underset{8}{\boxed{\begin{matrix}0&0&0\\0&0&0\\0&0&0\\0&0&0\\0&.5&.5\\0&.5&1\\0&.5&.5\end{matrix}}}$$

Figure 16. Gray patterns used to design the V2 feature detectors, as described in the appendix. Patterns 1 to 4 correspond to the presence of an edge, while patterns 5 to 8 correspond to a nonedge.

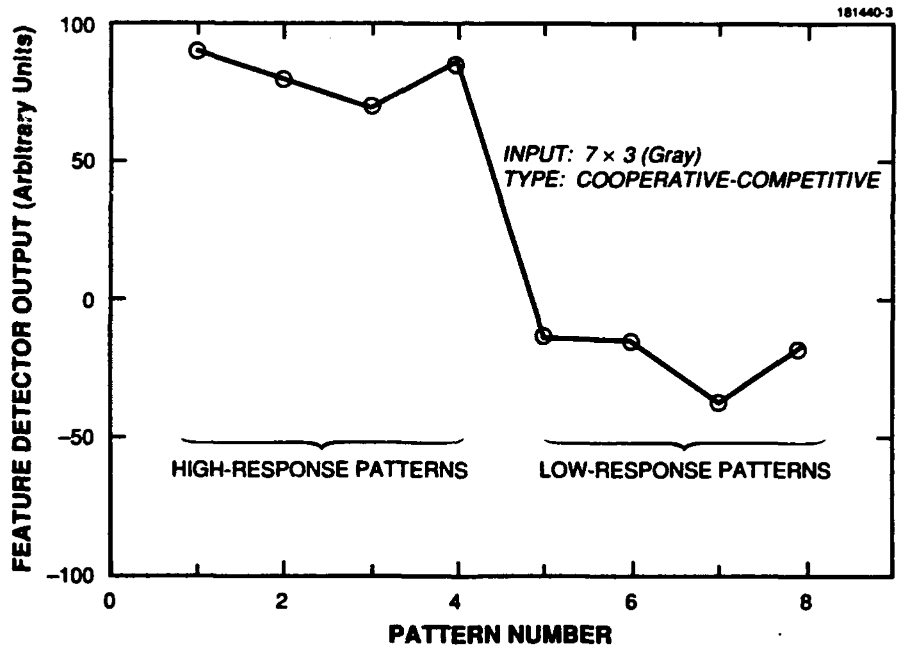


Figure 17. Response of the vertical WEST feature detector to the design patterns. For the baseline system, when an edge is present in one side of an image, the response is high. Otherwise it is low. The "signal-to-noise" ratio is at least 50 for the design images.

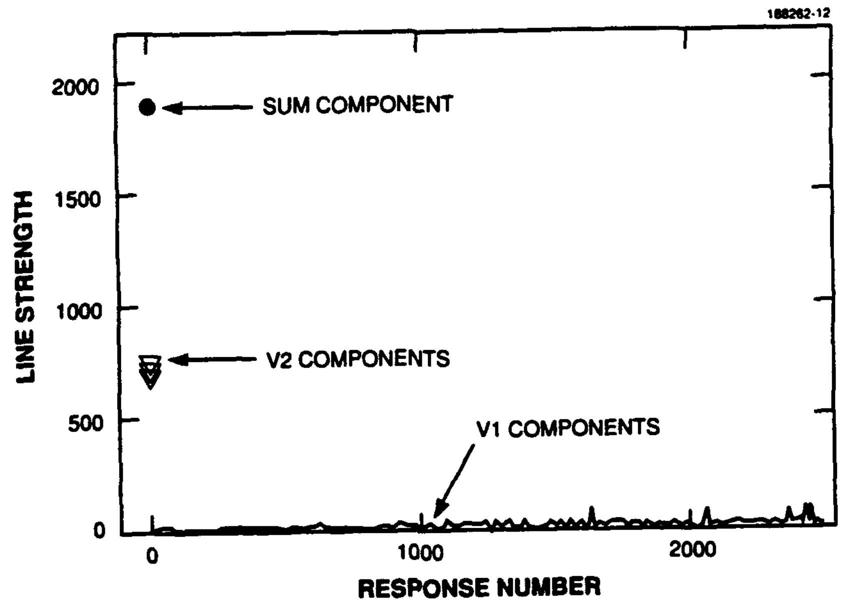


Figure 18. Feature vector of a normal Pap smear cell. The three kinds of components (SUM, V2, and V1) of the feature vector are weighted so that classification is based on size (SUM), general shape (V2), and interior details (V1).

188282-13

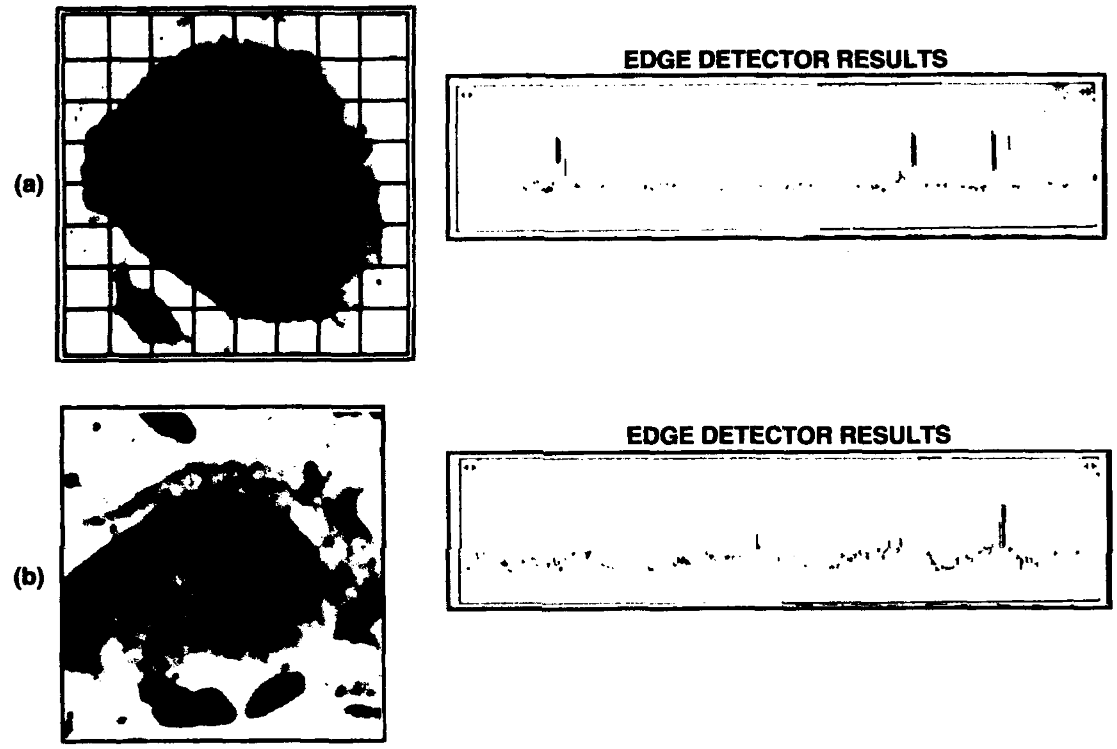


Figure 19. Comparison of (a) normal Pap smear and (b) abnormal Pap smear cells. Shown on the right are 512 of 2500 V1 features from the center of the images, that is, near the nucleus. The grid (not to scale) suggests the 7 × 7 boxes of V1 within which edges are detected.

4.2.6 ITC1 Module-Unsupervised Classification

The recognition process consists of an unsupervised classifier (ITC1) followed by a supervised one (ITC2). For the unsupervised classifier, the well-known ART-2 NN was used. ART-2 was selected over other NN classifiers, such as perceptrons and Hopfield NNs, because of ART-2's speed, stability, feature amplification, and noise reduction — characteristics that were better suited to the application in this study. And, ART-2 is a better model of the biology (for a discussion see Carpenter and Grossberg [17]).

Adaptive Resonance Theory (ART) is a learning theory introduced by Boston University Professors G. Carpenter and S. Grossberg [17]. ART mimics the human brain by taking inputs from the environment, organizing the inputs into internally defined categories, and then recognizing similar patterns in the future.

There are three classes of ARTs. ART-1, which was developed first, is used with binary inputs; ART-2 is used with patterns consisting of real numbers; and ART-3 handles sequences of asynchronous input patterns in real time. This study used ART-2. Several versions of ART-2 exist, but they all have the same basic characteristics.

Figure 20 shows the basic structure of ART-2, an NN with two levels of interconnected neurons, F1 and F2. The neurons are mathematical models of biological neurons. In the figure, the bottom layer of F1 receives the input pattern — a list of numbers representing the input. F1 consists of three layers of interconnected neurons that filter out noise, enhance the shape of the pattern, and rescale the input pattern values. The filtered pattern appears at F1's top layer, which is connected to the F2 level. The filtered pattern, called the exemplar, is the pattern that ART-2 stores.

In the F2 level, each neuron represents a category, or, with high sensitivity, one example input that defines a category. The activation of F1 and F2 nodes models the activation of biological neurons.

The F1 and F2 levels are connected in both a bottom-up and top-down fashion by the long-term-memory (LTM) trace. Mathematically, the LTM trace is the set of weights given to the F1 nodes as they attempt to turn on an F2 node by a winner-take-all competition. Functionally, the LTM trace stores information permanently or until the trace is modified by learning. The LTM models the synaptic junctions of biological neurons.

To train an ART-2, the LTM trace values are set according to a rule given by Carpenter and Grossberg [17]. Next, training patterns are presented to F1 one after the other. Initially, when ART-2 is untrained, the first pattern immediately causes the NN to enter into the learning mode. The network learns the pattern by modifying the weights associated with one of the F2 nodes.

After the first pattern is learned, each succeeding pattern will trigger the network to search for a match among the F2 nodes. If the pattern is a close match to a previously learned one, ART-2 enters the learning mode and modifies the LTM trace so that the trace represents a composition

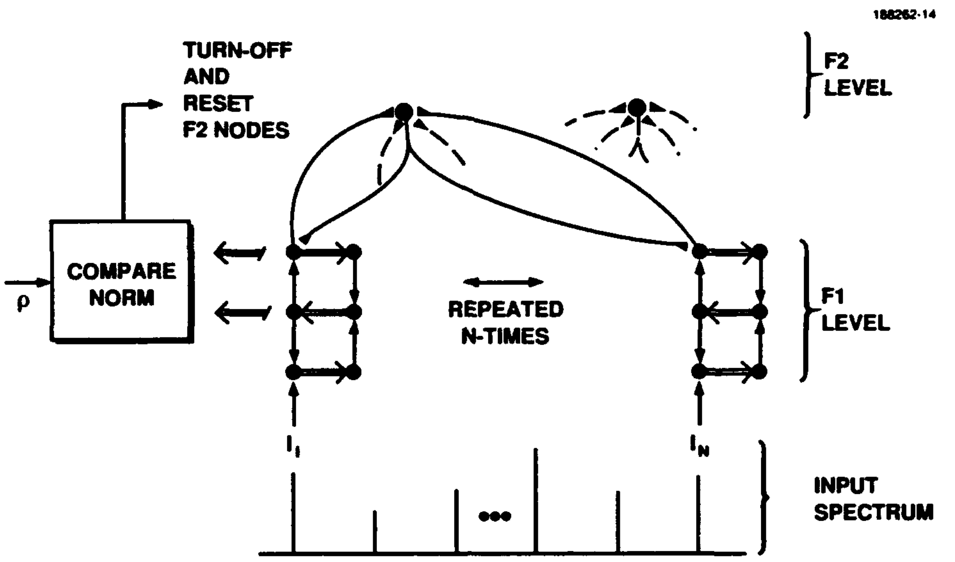


Figure 20. Summary of the ART-2 classifier, a neural network with two levels, F1 and F2, that consist of interconnected neurons. The bottom layer of F1 receives an input pattern (feature vector) that is then filtered, enhanced, and rescaled by the three F1 layers. The filtered pattern appears at F1's top layer, which is connected to the F2 level. The filtered pattern (exemplar) is the pattern that ART-2 stores.

of all the past, closely matched patterns. If the pattern is mismatched with all those previously learned, ART-2 goes into the learning mode and modifies the weights associated with an unused F2 node. Thus each pattern is automatically associated with an F2 node, and in this way ART-2 programs itself.

After training is completed and a new pattern is presented, the pattern's exemplar is produced. ART-2 then searches the LTM trace for the most closely matched exemplar. When a match is found, the corresponding F2 neuron turns on, indicating the category that best matches the pattern.

Before using ART-2, parameters must be set that influence the network's performance. For many of these parameters, suitable values have been determined by experience. One parameter of particular importance is the Vigilance, which serves as a threshold on the degree of similarity between the LTM trace and the input pattern's exemplar. If a certain mathematical matching formula equals or exceeds the Vigilance, that pattern will be associated with the corresponding F2 node. When the Vigilance criterion is not satisfied, ART-2 declares a mismatch and searches for a match among the other nodes.

The selection of a low Vigilance value (that is, a value near zero) leads the system to tolerate large differences, resulting in coarsely defined categories. A high Vigilance value (that is, a value near one) leads to increased sensitivity in pattern discrimination, resulting in finely defined categories. In practice, the Vigilance should be adjusted high enough to distinguish patterns that represent different categories. The value, however, should be low enough that slight changes resulting from incomplete or wrong information will not cause misclassification.

4.2.7 ITC2 Module-External Names

After training, the ITC1 (ART-2) output nodes in F2 correspond to particular patterns, or objects. For example, if the first ten training images are tanks, the first ten ITC1 output nodes will correspond to tanks. In the basic system, the supervised classifier, ITC2, is a simple logical OR operation to associate activity of these nodes with the name TANK. (Note: ITC1 is called an unsupervised classifier because the label of an input pattern is the F2 node number, which is internally and automatically defined by the algorithm. ITC2 is called a supervised classifier because the user defines the labels.)

After ITC2 processing, the system decides whether to store the object's name and location, and the ART-2 matching parameter [17] serves as a confidence measure for the decision process. If the matching parameter just passes a threshold (the Vigilance), the "confidence level" is 50 percent. A perfect match corresponds to a "confidence level" of 100 percent. If the "confidence level" passes a second threshold specified by the user, the system stores the results. But if the level is not sufficiently high, the location channel adjusts the window (discussed in the following section) and the system processes the image again.

4.3 Location Channel

The location channel places an input window around an object so that the system might classify it. As shown in Figure 6, the location channel consists of the following modules: superior colliculus (SUPERC), LGN, V2, and posterior parietal cortex (PPC). Location is a two-stage process consisting of coarse location followed by pull-in.

In practice the location processing is as complex as the classification processing. It must quickly find patterns of interest in the background clutter. Location also is as important as classification. For example, the search for one or two abnormal cells in 50,000 (typical of a Pap smear slide) is a location problem. After finding an abnormal cell, recognition is relatively easy.

4.3.1 SUPERC Module-Coarse Location

SUPERC processing, shown in Figure 21, uses a second ART-2 NN to perform coarse location. The network's LTM trace, which is computed off-line, corresponds to general shapes of interest. This trace primes the system. To detect the presence of an object, the SUPERC ART-2 compares the exemplar of its current window to the LTM trace. Even an off-center object will trigger a match if the object's size is correct.

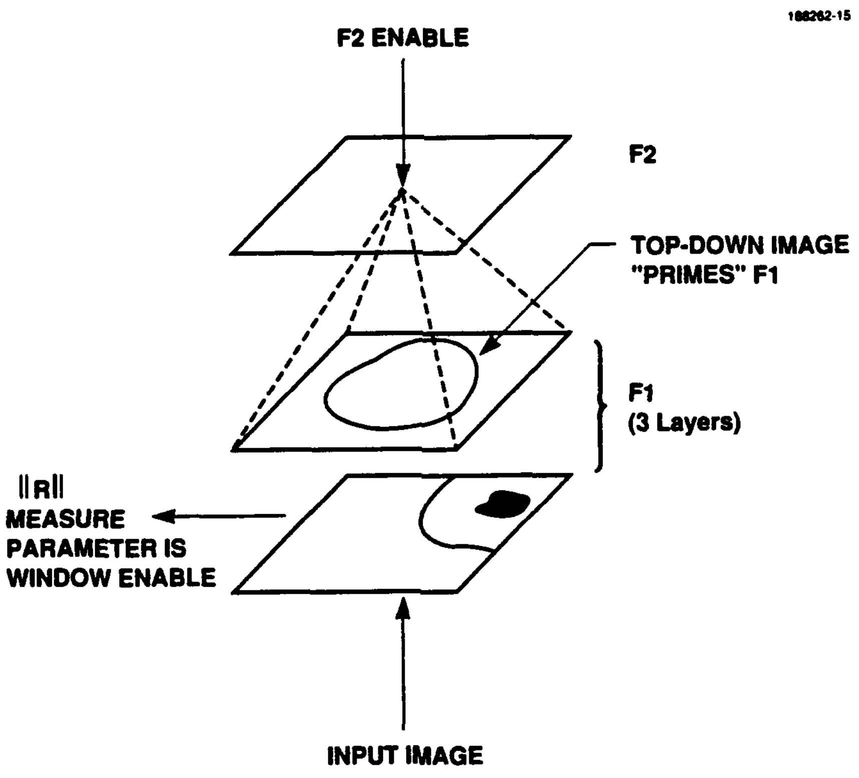


Figure 21. Schematic of the coarse location processing. A stored top-down exemplar in an ART-2 neural network primes the system for detecting objects of general size and shape, even if off-center.

In the example, SUPERC extracts a 175×175 pixel window from the input image. It impresses the window on the F1 bottom layer. The 175×175 pixel images produce 30,625 inputs to the ART-2. The exemplar is computed. The system compares the LTM trace to the exemplar. The designer selects the LTM trace so an object of the correct general size causes a match, even if off-center. A match indicates an object of this size is present.

If the system finds no match in a window, it moves on to an abutting window. In the example with a 525 × 525 FOV, there are nine coarse location positions. The system uses the match parameter as an Enable signal to the LGN module. A module inside SUPERC selects the coarse window positions. The system sends the coarse position to the ADJ box for further adjustment (Figure 6).

4.3.2 PPC Module-Pull-In

The second stage of location is pull-in, or fine adjustment of the coarse location. Pull-in operates on a feedback path consisting of the LGN, V2, and PPC modules. Using the outputs of V2, PPC makes small changes in the window's position. When the system centers a window on an object, all the V2 edge strengths are about equal. Otherwise, PPC tries to equalize the V2 edge strengths.

For example, Figure 22 shows the V2 output for an object wholly (top) and partially in the window (bottom). When an object is centered in the window, the edge strengths will be about equal. If the object is not centered, that is, it is partially in the window, one or more edges will be missing. Then, the corresponding edge strength will be small. The system moves the window to equalize the edge strengths.

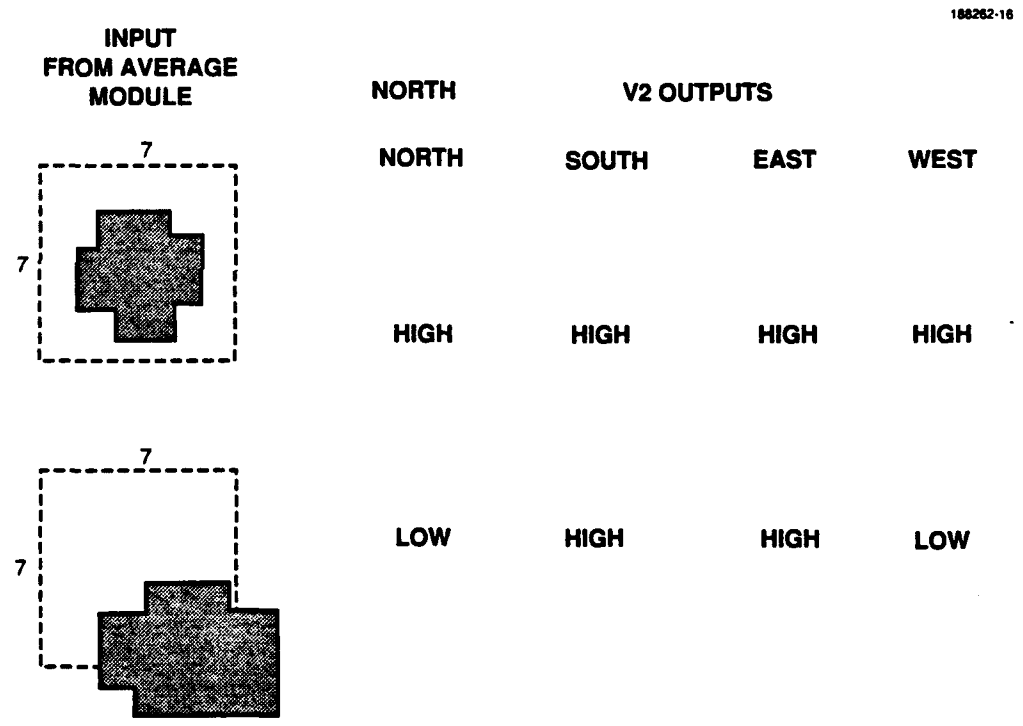


Figure 22. Examples of (a) the window centered on an object and the corresponding four V2 outputs, and (b) the window not centered. When the window is not centered, the unequal V2 outputs produce signals that move the window (pull-in).

To center the window on an object, the system routes the edge strengths from V2 to the DELTA1 box in PPC. This box carries out a control law for moving the window. For example, Figure 22 shows an object that is below and to the right of the window. The position produces a smaller north than south response and a stronger east than west response. To center the object, the DELTA1 (Figure 6) box must move the window south and east.

In the baseline design, the control law is a standard bang-bang rule with a dead-zone for the vertical and horizontal directions. The output of the DELTA1 box is the adjustments in the vertical and horizontal directions. Figure 23 shows the control law used in the baseline.

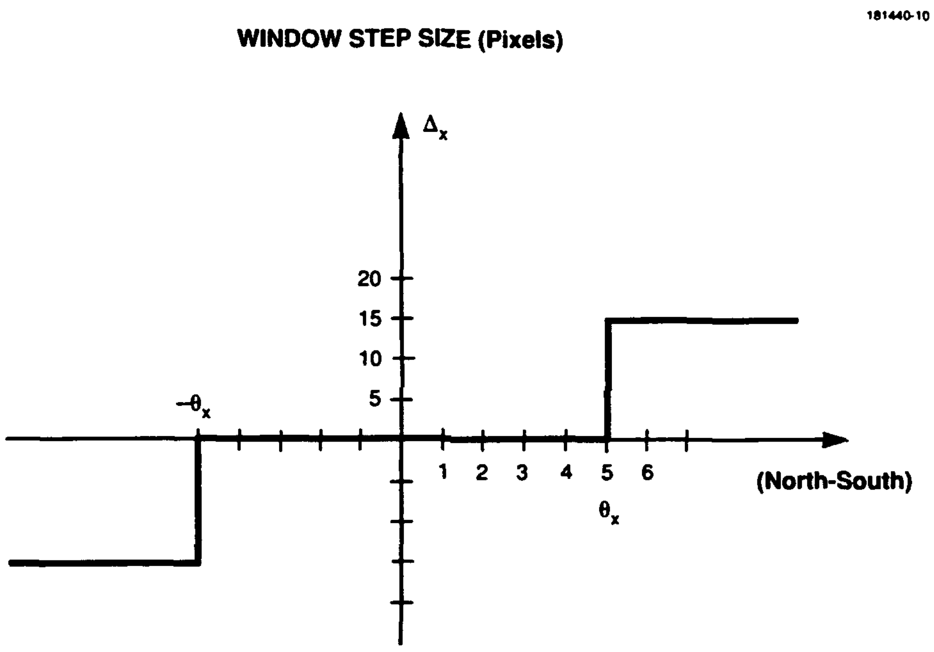


Figure 23. The baseline control law for vertical pull-in of the window. The two V2 outputs, NORTH and SOUTH, are subtracted and the window is moved if the thresholds are exceeded. In practice, the thresholds and step size are selected according to the application. A similar system with the EAST and WEST V2 outputs is used for horizontal pull-in.

A second pull-in path, which consists of LGN, V2, ITC1, ITC2, and PPC, makes repeated tries at recognition. ITC2 activates this path when the classification channel has low confidence in a match between an input pattern and the closest stored pattern. When the path is activated, the

DELTA2 box produces a small, random adjustment of the window's position and the system then tries to classify the object with greater confidence. A counter limits the number of tries.

4.4 System Dynamics

In computer simulations, the system executes sequentially. First, the location channel finds and windows an object. The classification channel then does identification.

In a parallel implementation with custom hardware, the modules run simultaneously. Sequence control is by Enable signals and carefully chosen time constants. Time constants associated with the location channel are short, so the system will converge quickly to the location. The classification channel time constants are longer and the identifying process is comparatively slow. The difference in the time constants ensures classification on a centered object.

5. THE SOFTWARE TESTBEDS

5.1 Introduction

A major goal was to test the architecture with computer simulation. To that end, a series of software testbeds was developed to study algorithm performance. As the testbeds were programmed to handle more complex and more extensive data, the architecture and algorithms evolved.

Table 3 summarizes the software-related aspects of this project. We built up three versions of the software. The first version was written in the APL*PLUS programming language and ran on an IBM PC/AT. Using synthetic binary images, such as alphabetic letters, the APL*PLUS software tested the algorithms of selected modules.

TABLE 3

A Summary of Software Issues

Testbed Issue	IBM PC/AT	Sun 4/110 and SPARC	Sun-CONVEX
Programming Language	APL*PLUS	C	C
Operating System	DOS	UNIX	UNIX
Input-Output Format	APL Native Files	STD and NATO	STD and NATO
Interface Graphics	None	Sun View	VIEWPROC (Calls SunView or X-Window)
Vectorization	Not Applicable	Not Applicable	Yes
Batch Mode	No	Yes	Yes
Image Types	Simulated	Simulated and Real	Real
Classification Channel Testbed	V1 and ITC1 Only	Yes	Yes
Location Channel Testbed	Coarse Only	Not Tested	Pull-In

environment for this kind of user interface through the SunView window management system. This graphical interface package provides simple methods for displaying results and controlling program execution. Such utilities minimize programming.

The classification channel testbed, called Cellview, creates and arranges a set of interaction windows. Figure 24 shows the resulting screen display. The user enters data through the keyboard and selects run options with a mouse. Text and graphical displays provide feedback about algorithm results.

The upper left window selects and controls the input images. To specify an image, the user enters appropriate information in text fields defining a directory, file name, and image number.

At the bottom of this window are several buttons to control the image display. To exercise these functions, one moves the cursor to an appropriate button and clicks with the mouse.

The "Display" button activates the testbed's image acquisition functions. An initial routine opens the specified image file and checks the file format. Subsequent routines locate the designated image, retrieve it, and produce a display. The display procedure provides descriptive information in several other fields. One of these fields identifies the display's color look-up table. Others show the magnification (zoom factors) and the position of a display window cursor.

The upper left window also has another button that clears the image display window. "Zoom" and "Unzoom" buttons change the magnification by a factor of two. A "Quit" button stops the program and exits.

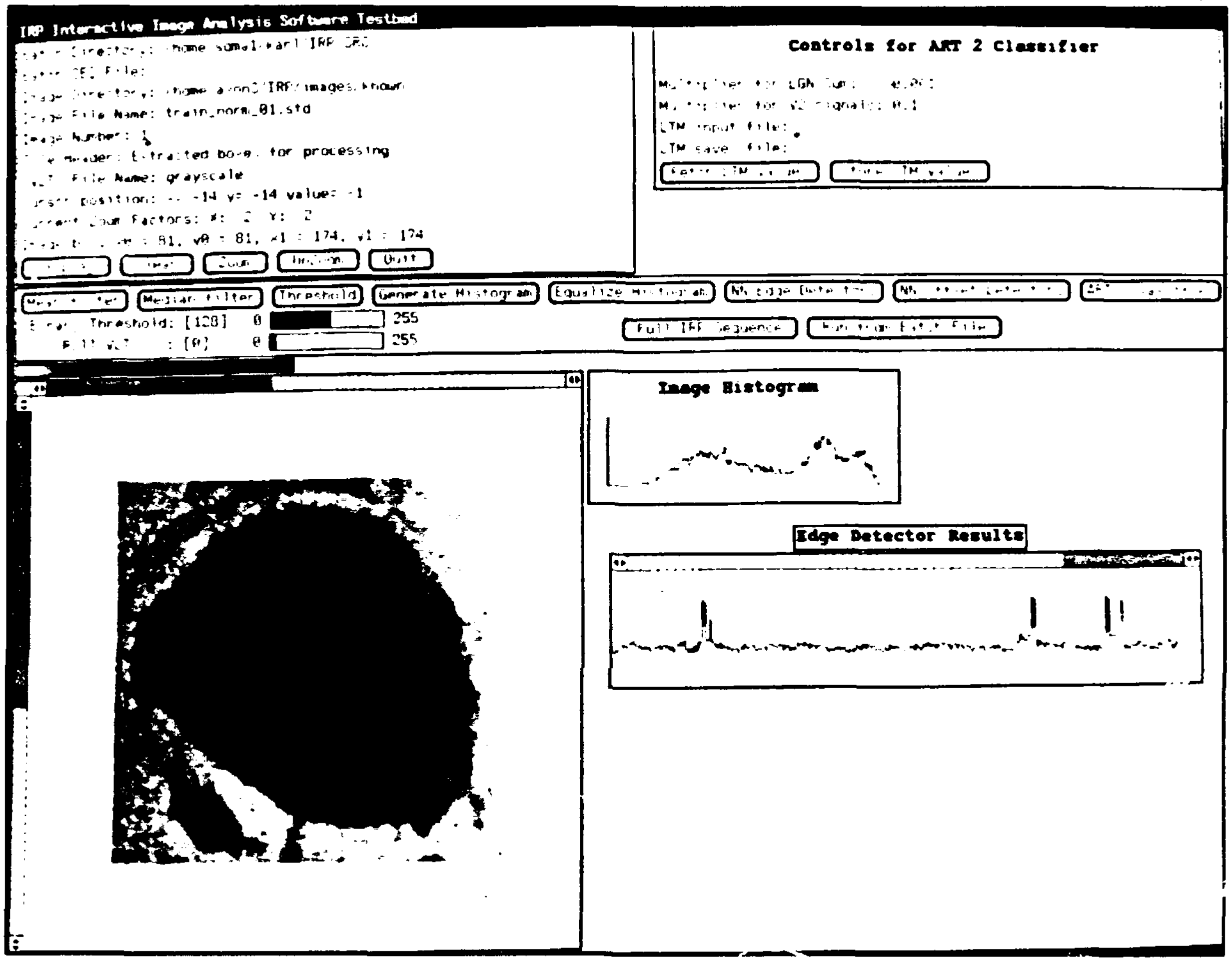
The lower left window displays the original input image and a modified version after processing. A video look-up table (VLT) determines the coloration by associating specific red, green, and blue intensities with each data value. The testbed gets this table from a file and the user can change the mapping by entering the name of a different VLT file.

The upper right window controls the ITC1 module's ART-2 classifier. The first two fields control the weighting of the SUM and V2 feature components. Default values are set in the testbed code. Other ART-2 parameters, such as Vigilance, reside in a parameter file. The last two fields accept file names for storing and retrieving the LTM trace. After entering the name of a target or source file, the user clicks a button at the bottom of the window.

A central horizontal window lies beneath the image control and ART-2 control windows. Buttons in this window run different parts of the algorithm. Two of these commands run the image histogram and perform histogram equalization. Others run the V1 edge detectors, the V2 edge detectors, and the ART-2 classifier, as shown.

The lower left and lower right windows display graphical results to assist with monitoring. After histogram equalization, the testbed shows a modified image in the lower left window. This display overwrites the original input image. Plots of the histogram and V1 features appear in the lower right, as shown.

141455-2C



[illegible] Directory: [illegible]home[illegible]IRP[illegible]
[illegible] File:
Image Directory: [illegible]home[illegible]IRP/images[illegible]known
Image File Name: train_norm_01.std
Image Number: 1
[illegible] Header: Extracted box[illegible] for processing
[illegible] File Name: grayscale
[illegible] position: [illegible] -14 y: -14 value: -1
[illegible] Zoom Factors: X: 2 Y: 2
[illegible] = 81, y0 = 81, x1 = 174, y1 = 174

[illegible] Threshold: [128] 0 255
[illegible] : [0] 0 255

Multiplier for [illegible]: [illegible]
Multiplier for [illegible] signal: 0.1
LTM input file:
LTM save file:

Figure 24. Sun workstation display of a Pap smear cell, its histogram, and its feature vector.

With these buttons and displays, a user can execute the algorithm step-by-step and inspect intermediate results. At every step in the algorithm, several options are available. For example, one can proceed to the next step or repeat earlier ones with a different image or other parameter values. The user also can carry out repeated tests while varying the parameter values or can stop the program at any time. The broad choice of options, commands, and feedback was a design goal.

Figure 25 shows the operator interaction during a typical test run as a flow chart. The interaction was suggested by the SunView button syntax. The flexibility of the interface greatly simplified the task of algorithm assessment, and, in practice, proved to be extremely convenient.

While coding and debugging the Sun testbed, we focused entirely on the algorithms and the user interface. During this period, the CONVEX minisupercomputer was installed and brought on line. Then the Sun-CONVEX testbed was developed. This software extension improved the run speed while preserving all previous capabilities.

In the Sun-CONVEX testbed, functions are distributed between the two host computers. A Sun workstation still controls the flow of data, and in this capacity, it performs I/O and interacts with the user. These functions are identical to those of the Sun testbed. The CONVEX carries out the most computationally intensive algorithms, which are V1 and ART-2. The separate subprograms communicate over Group 22's local area network.

To coordinate the Sun and CONVEX operations, a protocol for passing data and calling algorithm procedures was defined. The two computers use the TCP socket facility to establish a common communication channel. A server program on the CONVEX accepts input from the Sun. It then runs V1 or ART-2. After running one of these modules, the server sends the results back to the Sun.

The Sun programs were modified to communicate with the CONVEX. Commands were added to carry out steps in the processing sequence. Certain commands send image data to the CONVEX and compute the V1 feature values. Another command transfers the V1 results back to the Sun. The Sun combines the V1 values with the SUM and V2 components and performs weighting. Other commands send the complete feature vector to the CONVEX and run ART-2. Results from ART-2 are then transferred back to the Sun, which includes the index of the active F2 node, the number of iterations needed for learning, and the matching metric's value when it passed Vigilance.

When operating with the CONVEX server, the Sun executes these commands instead of its own V1 and ART-2 procedures. Our modifications, however, preserved the Sun's ability to run V1 and ART-2 on its own. A user chooses one of these options during the start-up procedure.

Two extra steps are needed to run using the CONVEX. First, one logs on to the CONVEX and starts the server program. Then the user appends a string "-convex" on the command line for starting the Sun program. This second step directs the Sun to work with the CONVEX. If either step is omitted, the Sun testbed runs alone.

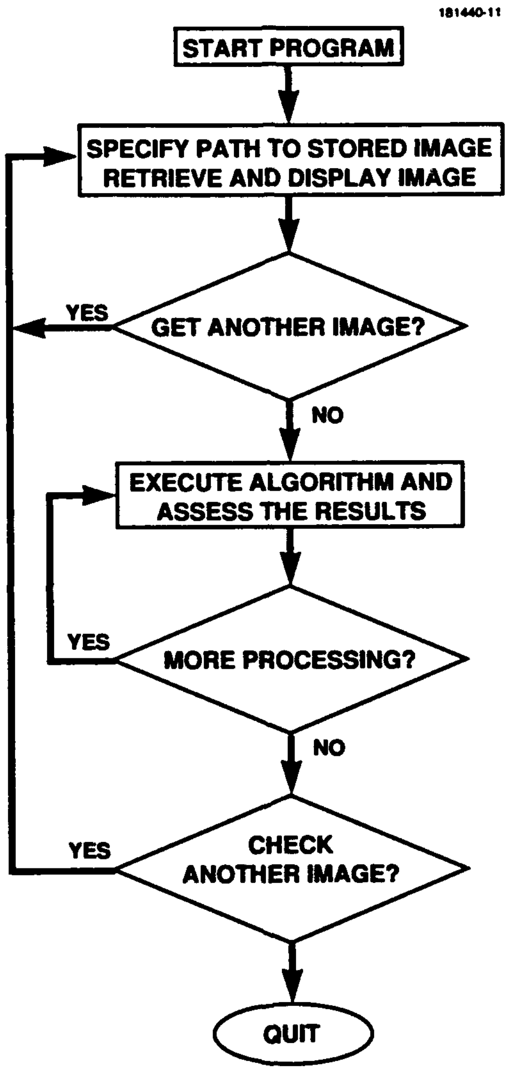


Figure 25. Flow chart of testing procedure.

5.3.2 Location Channel

Foveation is critical to recognition. In the system, foveation means finding a pattern of interest and centering a box or window on it. The approach of this study consists of two steps: coarse location followed by pull-in, as described in Section 4.

Coarse location is easier than pull-in; preliminary tests of the course location were performed on an IBM PC/AT. Section 6 describes the tests. Pull-in needs more care, because reliable recognition depends on having the window well-centered on an object. The pull-in facility also has to work under real conditions, that is, in scenes with nearby objects and cluttered backgrounds.

A software testbed for the pull-in modules was designed and tested. The system runs on either a Sun workstation or a CONVEX minisupercomputer. In practice, speed usually favors the use of the CONVEX.

The testbed relies on two software packages developed in Group 22. One of the packages is the "opt" utility, which allows program parameters to be specified either interactively or from the command line with little programming effort.

The second software package is the VIEWPROC utility that produces and displays graphics output on a Sun workstation. VIEWPROC is designed to allow a graphics display device to be designated as any workstation in the network and can run on either the Sun workstations or the CONVEX system.

The location channel pull-in testbed maintains two display windows, shown in the next section (Figure 27). One window displays the image to be processed. A box is drawn in black at the initial position, given by the operator. As the box coordinates are modified, the color changes to brighter red to track the behavior of the pull-in over time. If the final position is reached before the maximum number of iterations have elapsed, the final position is drawn in green to highlight this fact.

The second window displays the portion of the image contained in the initial boxed region. Next to this image is the contents of the box at the current position. The final state of this window shows the initial and final box contents.

In practice, a user can quickly exercise the pull-in function over a wide range of conditions. During testing, one varies the initial position and other parameters with the selected image.

The test images can be natural or synthetic. In the course of this project, a software utility was developed to construct test images. The program allows extracting 175 x 175 images and inserting them into larger scenes of 525 x 525 pixels. For example, the 525 x 525 scene might have a plain, clutter-free background. Several extracted images also could be inserted into the same 525 x 525 frame. With this software, one can develop precisely controlled tests of the pull-in function.

5.4 Vectorization

To take advantage of the CONVEX's parallel-vector architecture, the V1 and ART-2 routines were modified. Even without these modifications, the CONVEX provides some benefit because each of its two processors performs arithmetic operations at about four times the speed of a Sun 4/110. With well-written software, however, this minisupercomputer can reach much greater speeds when executing loops.

During loops the program takes a large array of data and repetitively carries out a calculation. The CONVEX accelerates loops by vectorization, a technique in which separate arithmetic and memory management units concurrently perform different suboperations using pipeline hardware.

Each unit loads its inputs, performs a designated operation, and passes the result on as input for the next unit. On each machine cycle, one set of operands enters the pipeline and one final result goes to memory. The units, which feed the pipeline, manipulate data as vectors. In general, to be worthwhile, a pipeline must engage a significant portion of these resources and keep them busy a large part of the time.

The CONVEX C-compiler converts many loops into machine instructions for one or more pipelines. The compiler's goal is to maximize the number of arithmetic operations done in each cycle. The CONVEX C-compiler increases its yield by performing certain *limited* manipulations of the source code.

Some kinds of calculations, however, simply cannot be vectorized, for example, recurrence, which is a significant problem. Recurrence take places when input for a loop calculation is computed during an earlier iteration of the *same* loop. If the CONVEX *really* performed all loop iterations simultaneously, it could not compute the recurrent input values before they were needed, because the calculations are inherently serial. In general, pipeline timing depends on the detailed structure of a loop. With recurrence and with variations in the iteration time, the CONVEX cannot ensure that the results will be ready in time for later input.

The CONVEX C-compiler actively guards against this scheduling problem. While inspecting source code, it checks for recurrent expressions, and whenever the compiler finds one of these statements in a loop, it translates the statement into scalar machine instructions. The mechanism is conservative, that is, it bypasses vectorization whenever there is any possibility of recurrence.

Much of the V1 and ART-2 code is considered to be recurrent by the CONVEX C-compiler. This apparent recurrence occurs because the C programming language represents each array as a variable holding the memory address of a contiguous data block (a "pointer"). After defining an array variable, a program can freely change the specific address stored there. Thus, stipulated C code can arrange for distinct array variables to point at different elements in the same physical data block. When this situation occurs, a loop will be recurrent if it sets values and takes inputs in those arrays.

The CONVEX C-compiler can check for these recurrent array references, but only when the array addresses are strictly local to a single C subroutine. Such checking becomes unreliable when the actual array addresses pass among different subroutines as arguments or global variables. When software modules share array pointers, any subroutine can assign the array addresses. Thus, at any point in the program, the actual array addresses will depend on a preceding history of subroutine calls.

Moreover, the C-compiler cannot evaluate these historical effects, because the order of subroutine execution may be data dependent. So, to be cautious, the CONVEX C-compiler bypasses vectorization whenever a loop accesses arrays defined as global variables or subroutine arguments.

Indeed, most data arrays in the V1 and ART-2 code fall into the class of global variables. Thus, the CONVEX C-compiler flagged many expressions as recurrences and declined to vectorize them. Fortunately, a programmer can override the antirecurrence mechanism on a statement-by-statement basis, by placing the string

```
''\$dir no_recurrence''
```

immediately before a loop. In this way the C-compiler is forced to vectorize, in spite of an apparent recurrence. The directive is entered as a C-language comment, and the programmer must make certain the expression actually is nonrecurrent before overriding.

All the flagged recurrences were examined; most were false. By overriding and vectorizing these loops, the speed of the V1 and ART-2 computations was greatly increased.

While a few of the flagged recurrences were genuine, most could be broken down into a sequence of nonrecurrent steps. This reformulation replaces a single recurrent loop expression with a sequence of two or more loops. The new sequence of computations produces the same result as the original formula. This technique was used to cut out all but one of the true recurrences in the ART-2 programs, thus increasing the proportion of vectorized loops.

Vectorization of the V1 and ART-2 algorithms yielded disparate benefits. For ART-2, run time with a 175 x 175 image dropped from about 20 min on a Sun 4/110 to about 18 s on the CONVEX. For V1, the gains were substantial, but less dramatic: 4.5 min on a Sun 4/110 to 45 s on the CONVEX.

The disparity of benefits arises because of the different computational structures of the algorithms. For V1, matrix multiplication and sigmoid functions lie at the heart of the algorithm. To compute responses for the NN's hidden units, the algorithm iterates these operations in a series of successive approximations, described in the appendix. The procedure decides when to stop by comparing the current response values with those from the preceding iteration. It halts when the differences fall to zero.

The hidden unit calculation in V1 is highly recurrent because results from each iteration serve as input for the next. The termination criterion prohibits vectorization because the actual number of iterations is uncertain. In general, pipelines must carry out a fixed sequence of arithmetic operations and cannot handle this conditional loop control. Thus, in V1 only the *individual* matrix multiplication and sigmoid operations can be vectorized.

To process an entire 175 × 175 scene, the V1 hidden unit calculation must operate 5000 times (625 positions × 4 orientations per position × 2 gradient directions per orientation). Each position requires an additional matrix multiplication and sigmoid to produce the final output signal from the hidden unit responses. On average, the V1 algorithm carries out many tens of thousands of matrix multiplications per scene. For each of these operations, the CONVEX must set up and shut down a separate pipeline.

Moreover, overhead is another consideration. Whenever feeding a pipeline, the CONVEX performs certain overhead tasks, such as loading data into a vector register and waiting for the first output to reach the end of the vector computation. Efficient programs can handle large amounts of input for each unit of overhead.

In the V1 algorithm, vectorized loops operate on relatively small data sets; the input window contains 49 pixels, and the hidden layer has 25 units. Thus, with a large number of these short loops, the V1 algorithm has high overhead per pixel.

By comparison, most loops in the ART-2 programs work on a feature vector with 2505 elements. The ART-2 algorithm also executes considerably fewer loops over its course of execution. Thus, the ART-2 code has a much lower ratio of overhead to throughput. These factors are responsible for the difference in performance gains between V1 and ART-2 (our colleague C. Mehanian helped to optimize the ART-2 routines).

5.5 Batch Mode

After developing the interactive program, a batch capability for testing large databases was added. In the batch mode, the program automatically reads input image specifications from a disk file. Within memory limits it can handle an arbitrary number of files. The batch option runs images through the full algorithm without operator intervention.

To run the batch mode, the user provides a Batch Directory and Batch File Name through text fields in the upper left window. These inputs identify a file containing the list of input images. The user then selects the "Run from Batch File" button in the central horizontal window.

As each image is processed, the batch routine records results in a log file. When the algorithm runs successfully, a log file entry gives the ART-2 class (F2) node, the number of iterations needed to learn the pattern, and a value for the matching metric when it first passed Vigilance. In unsuccessful cases where an error occurs, the program writes a descriptive message. In practice, faulty input image specifications are the most common cause of error.

To develop the batch mode, a different Group 22 software resource was exploited. This facility provides utility programs for handling lists of input images. These ASCII "sequence files" list the desired input images in a specific syntax. Individual entries give the full pathname (directory and filename) of a file and a single interval of consecutive frame numbers.

One of the utility programs reads the sequence file and formats the image specifications for internal use. Another utility program furnishes image specifications in the same order given by the file, as the processing passes from one image to the next. A user must create the sequence file before running the testbed program. The sequence file dictates the order of processing by composing the list of image specifications. The testbed's batch facility greatly shortens the testing time.

6. TEST RESULTS

6.1 Introduction

Current MV systems usually do not perform well on images from natural scenes. As discussed in Section 1, the reasons are many and well documented. The goal of this project was to develop a general purpose MV system. The method for testing the system was exercising the algorithm with two widely differing classes of images: vehicles and cells. These two classes were selected because large databases exist and because most researchers judge them significantly different.

Major subsystems were tested to assess their performance. Because of time constraints exhaustive testing was not possible, sensitivity studies were not conducted, and the system was not integrated. Moreover, to simplify the interpretation of the results, the location and classification channels were tested separately. Nevertheless, enough testing was executed to characterize roughly the performance of the two channels.

For the location channel, individual testing with different images was performed on the coarse location and pull-in functions. If the location channel does not function, the system will not center the image at the fovea, and classification performance will then suffer, perhaps significantly.

For the classification channel, the objects were centered, or foveated, by hand so that the tests evaluated recognition under ideal conditions. Thus, the results gave an upper bound on system performance because the location process introduces additional errors.

The preliminary test results were auspicious. The system located and recognized objects in their natural settings, and the algorithm was robust with respect to centering accuracy and background clutter. More testing is necessary before fully knowing the power and limitations of this approach. Moreover, massive testing is necessary for measuring error rates less than a few percent. One also needs to test other images besides cells and vehicles. [1]

Nevertheless, enough tests were run to suggest that this is a general approach to the MV problem. More than one general solution, however, may exist, and another design especially tuned for an application may give superior performance.

[1]Since completion, tests with laser radar range and SAR imagery have been performed. The system was trained on video and tested on laser radar range and vice versa. The software has been modified to include feature fusion. These results are reported elsewhere. They generally support the results discussed in this report.

6.2 Location Channel Tests

6.2.1 Coarse Location

Coarse location is the first step in foveation. The system coarsely locates objects of interest, for example, cells and vehicles in their natural backgrounds using an ART-2 NN. The process is usually called segmentation.

In the system, coarse location operates as follows:

1. The Fovea Move module drives the fovea to a new position (Section 4.3.1).
2. An F2 node is excited. A top-down image is produced on the top layer of F1 (priming).
3. The system compares the input image with the stored general images. If an object is present, the comparison parameter (norm) is larger than the Vigilance threshold.
4. The position coordinates and norm are inputs to the LGN module. The coordinates define the first box location. The norm signal is an Enable signal to the LGN module.

Three main questions arise about coarse location of objects in the FOV.

1. Can the system find all objects of interest?
2. What is the sensitivity to object shape?
3. What *Vigilance threshold is needed?*

To study these issues, a software testbed of the algorithm on an IBM PC/AT written in the APL*PLUS language was developed. Although this testbed was too slow for real images it was designed to test the concept by doing simple binary images and silhouettes.

Coarse location tests were performed on the PC testbed. The PC testbed simulated finding objects in the FOV. A 25 x 25 pixel FOV with 5 x 5 coarse locations was assumed. A 3 x 3 pixel pattern could be positioned at each coarse location.

An ART-2 was trained on a single, uniform 3 x 3 pattern at each coarse position. This gave 25 F2 nodes, one for each position. Figure 26 (top) shows the coarse position numbering. The figure shows a training pattern in the first position.

Six simple binary test patterns are defined. Figure 26 (bottom) shows these patterns. By experimenting, it was found that setting the Vigilance to 0.79 detects a test pattern.

In test patterns 1 and 2, at the positions shown, the system found all four objects. The norm of each object is 0.88, greater than the threshold of 0.79. At positions without objects, the norm is 0.75.

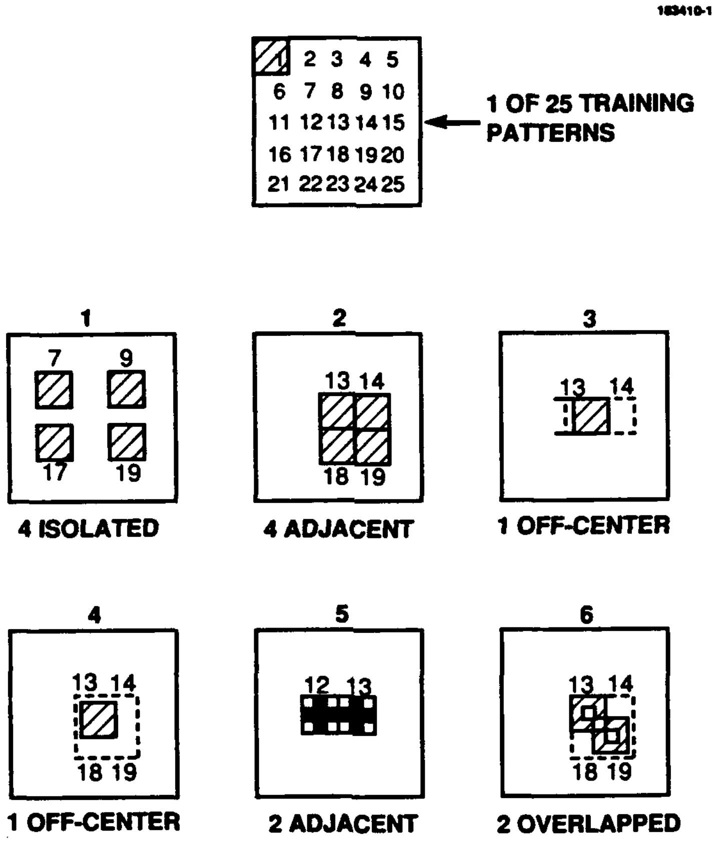


Figure 26. Training and test patterns for coarse location SUPERC module.

For test pattern 3, a 3×3 uniform object was translated one pixel to the right. The pattern lies partially at positions 13 and 14. The norms for positions 13 and 14 are 0.93 and 0.84, respectively. Thus, with the above threshold the system shows an object at the two positions.

For test pattern 4, a 3 × 3 uniform object was translated right one pixel and down one pixel from position 13. The norms for positions 13, 14, 18, and 19 are 0.88, 0.81, 0.81, and 0.78, respectively. Thus, with the above threshold the system shows an object at positions 13, 14, and 18, and none at 19.

For test pattern 5, the corner pixels of two adjacent objects were removed. Nevertheless, the norms at positions 12 and 13 are 0.88, specifying an object.

For test pattern 6, the center pixels of two overlapping objects are missing. The norms at positions 13, 14, 18, and 19 are 0.93, 0.80, 0.80, and 0.82, respectively. Thus, the system detects objects at all these positions.

For these simple binary shapes this approach works. Moreover, the algorithm allows priming the system for general shapes without recognizing them. The algorithm is robust to offset and noise, although more testing needs to be done, especially with real images.

6.2.2 Pull-In

Pull-in is the second foveation step. The algorithm was programmed on the Sun-CONVEX system. Pull-in has feedforward and feedback signals, described in Section 4.3.2. Section 5.3.2 describes the software testbed.

Pull-in was tested by producing test scenes. To create a test scene, a background was selected first. Then, one or more cells were selected and placed in the background.

To run a test, the box is positioned for windowing. In a fully integrated system, the coarse location channel positions the box. The software testbed then moves the box according to the procedure described in Section 4.3.2. The testbed records the position of the box at each step.

Figure 27 shows a typical result for a single cell in a natural background. The two boxes on the right show the first and last box positions. The left screen shows the box history. The first position (black) moves to intermediate positions (red) and then to the last position (green).

Repeated runs of the testbed gave the pull-in characteristics. Figure 28 shows the simplest occurrence, the pull-in characteristics of a cell on a plain background. For example, starting at $(-5,4)$ the trajectory of the pull-in is $(-4,3)$, $(-3,2)$, $(-2,1)$, $(-1,1)$, and $(0,0)$. Note that each increment is 15 pixels.

After many runs, the pull-in characteristics for this case can be summarized. At a radius of 75 pixels (five increments) from the center, the system pulls in and centers the box within 15 pixels (one increment) of the midpoint. For cell sizes of roughly 175 x 175 pixels (cervical epithelial cells at 400x magnification), this pull-in range corresponds to 42.8 percent of the cell diameter. Thus, if 42.8 percent or more of a cell's diameter is inside the initial windowing box, the pull-in aligns the box within 8.6 percent of the diameter in eight moves or less. Figure 29 shows a typical pull-in trajectory for a single cell on a plain background.

At greater offsets the system may oscillate. Several representative situations were tested. Figure 30 shows two cells in a natural background. The system pulled in the first box in six steps. But for some first positions, the system oscillates between two cells.

These tests show adequate pull-in range. The coarse location sensitivity is chosen so objects outside the pull-in range will be in the range of an adjacent box. The sensitivity to differences in the gradient across V2 remains to be studied. These and other situations, for example with overlapping, also remain to be studied.

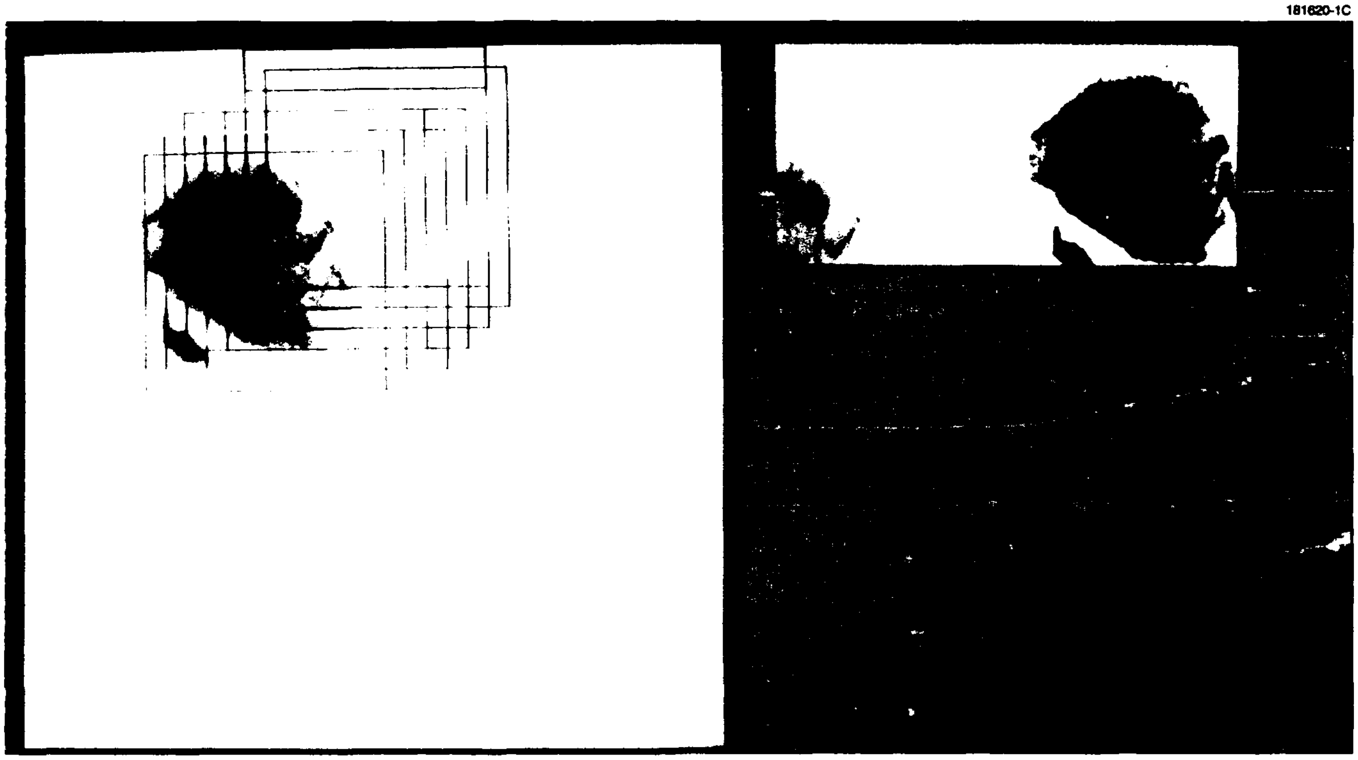


Figure 27. Sun workstation display of a Pap smear cell for a pull-in test. The two smaller images on the right show the initial (middle) and final (right) positions of the window. The left screen shows pull-in with the starting window position in black, intermediate positions in red, and the final position in green.

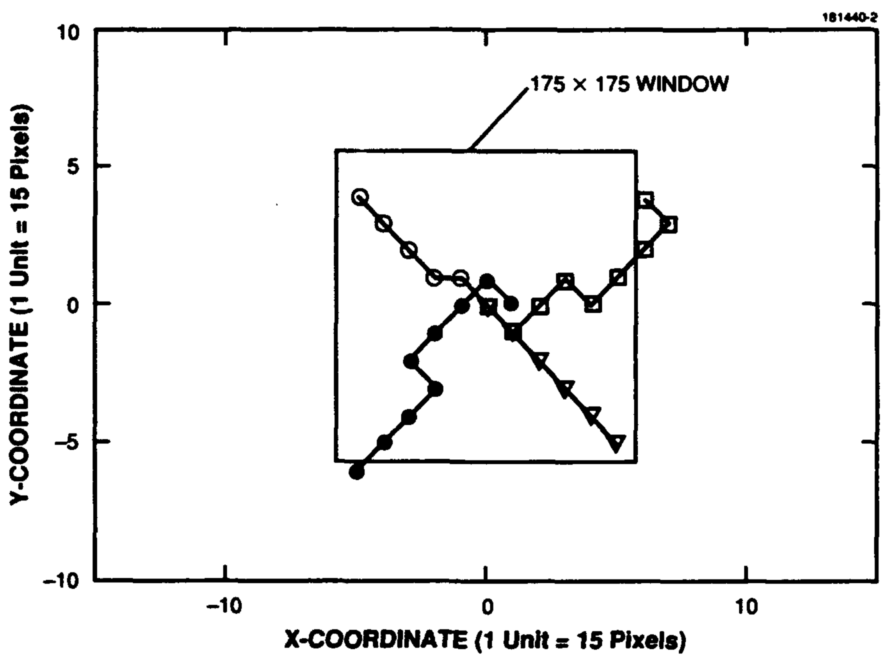


Figure 28. Example pull-in characteristics for a single Pap smear cell on a plain background. The figure shows pull-in from four quadrants starting near the periphery to the center of the 175 × 175 pixel window.

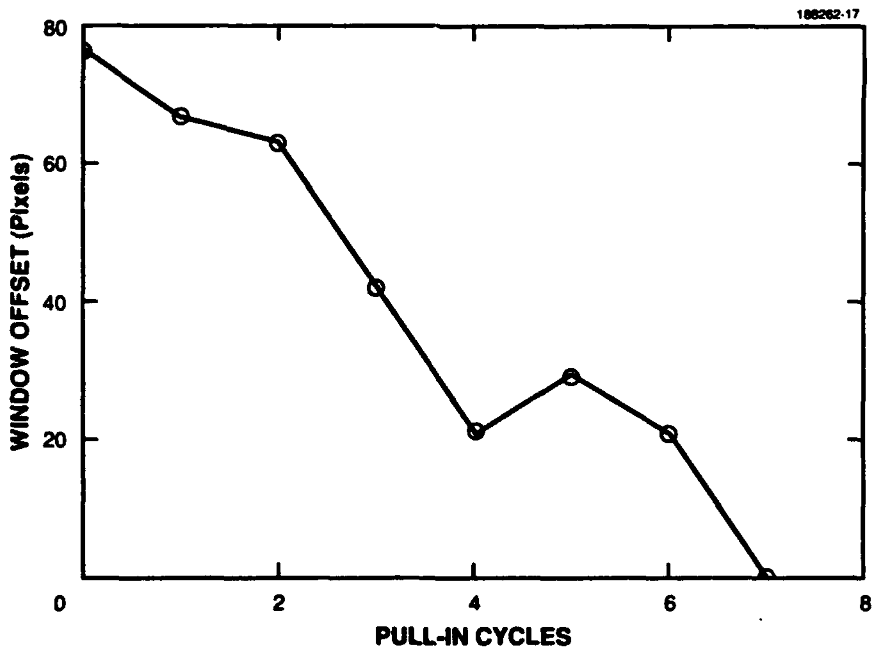


Figure 29. Typical pull-in trajectory on a single cell. Starting with an initial offset of the window, the pull-in system centers the window on the object so recognition may be done by the classification channel.

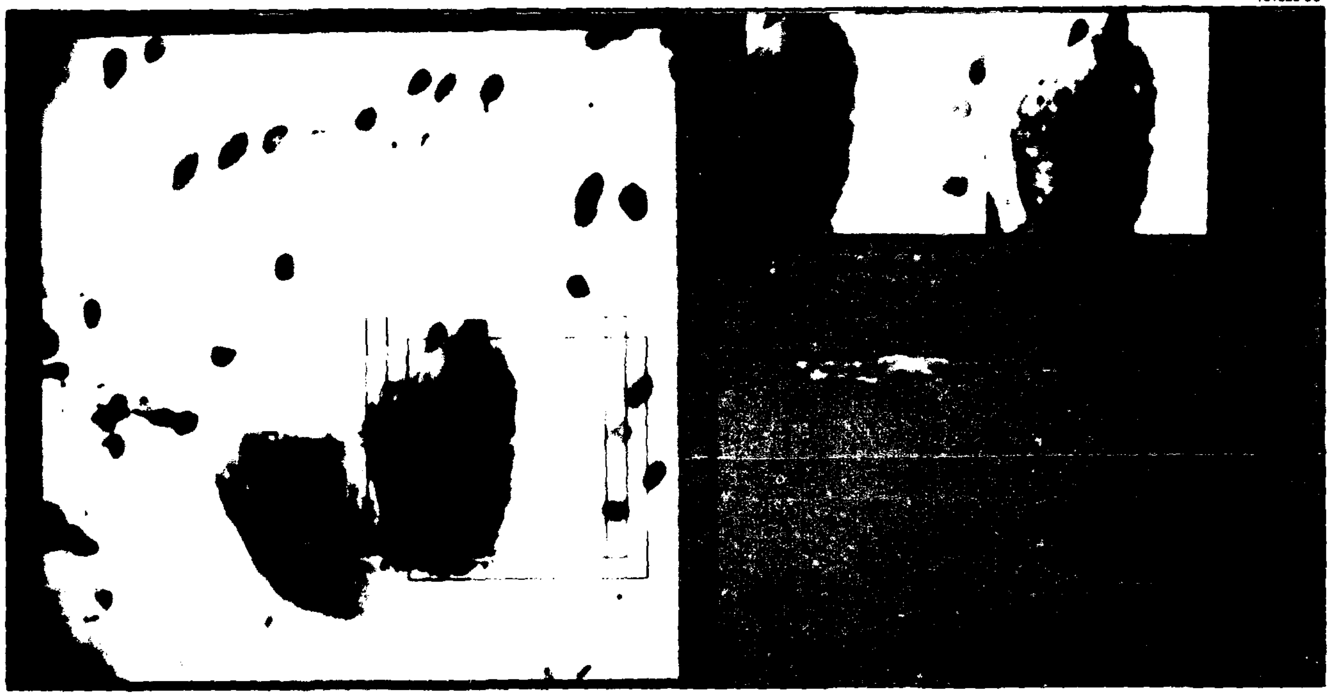


Figure 30. Example of pull-in on one of two cells in their natural environment. As before, the two smaller images on the right show the initial (middle) and final (right) positions of the window. The left screen shows pull-in with the starting window position in black, intermediate positions in red, and the final position in green.

6.3 Classification Channel Tests

To study the system's performance for recognition, tests were performed on two very different databases. This section describes tests with images of military vehicles and Pap smear cells.

6.3.1 Vehicles

A database of three common military vehicles is assembled: a M48A5 tank, a M113 APC, and a M110 self-propelled howitzer. The database consists of 40 images: 4 APCs, 16 howitzers, and 20 tanks. The vehicles were at 700-m range, with orientations that varied from front-on to broadside to end-on, and the background consisted of trees and rolling hills.

The images, intensity measurements made with a low-level TV camera, were 120 × 128 pixels with each pixel containing 8 bits, so that 2^8 gray values could be represented [20]. The beamwidth of each pixel is 300 × 300 square microradians. No effort was made to improve these images with, for example, preprocessing. Figure 1 shows a typical tank image.

To simulate the foveation of an image, a 42×42 window was centered by hand. This procedure measures the error rate caused by the classifying. The 42 × 42 pixel box enclosed all the images, regardless of orientation. All the boxes include background pixels, and at least 20 pixels in the minimum projected dimension were on the target.

The system then produced a feature vector that contained three kinds of data: SUM, V1, and V2. The relative weighting among the three components varied. Large SUM and V2 values resulted in V1 having little effect on recognition, while small *SUM and V2 values allowed V1 to play* a predominant role. By adjusting the SUM and V2 multipliers, the three feature-vector components are given roughly equal influence.

To find suitable weighting values, the system was trained on a small set of images and watched the resulting number of categories that ART-2 formed. Figure 31 shows the results for an 18-image training set.

When the SUM and V2 multipliers have low values, the number of categories is the same as the number of inputs. The system, however, becomes more sensitive to the interior details of the image, and to the noise. As the SUM and V2 multipliers increase, the V1 features become less important and the system loses its ability to discriminate between certain categories. Consequently, the number of ART-2 categories decreases below the number of input patterns.

The dotted line in Figure 31 represents a rough boundary for this transition. To obtain roughly equal weighting for SUM, V1, and V2, a point is chosen inside the boundary near the elbow as compromise between noise robustness and object discrimination. In general, the SUM and V2 multiplier values depend on the window size and sensor properties.

For comparison, Figure 32 shows the feature vector of a tank, a howitzer, and an APC. The system distinguished among these and similar feature vectors.

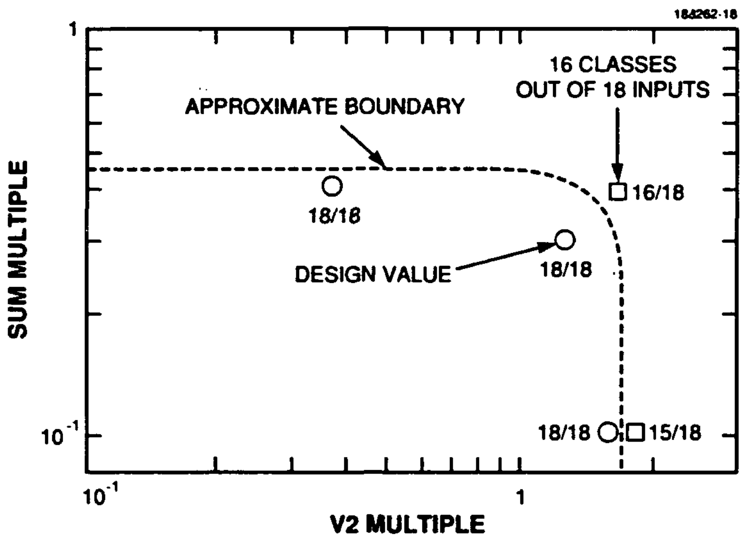


Figure 31. SUM and V2 multipliers for an 18-image training set of military vehicles. At low multiples the number of categories is equal to the number of inputs, but the system is more sensitive to noise. As the multiples are increased the system loses its ability to discriminate, and the number of categories decreases below the number of inputs. The dotted line represents a rough boundary for this transition. We selected a design value inside the boundary and near the elbow of the curve.

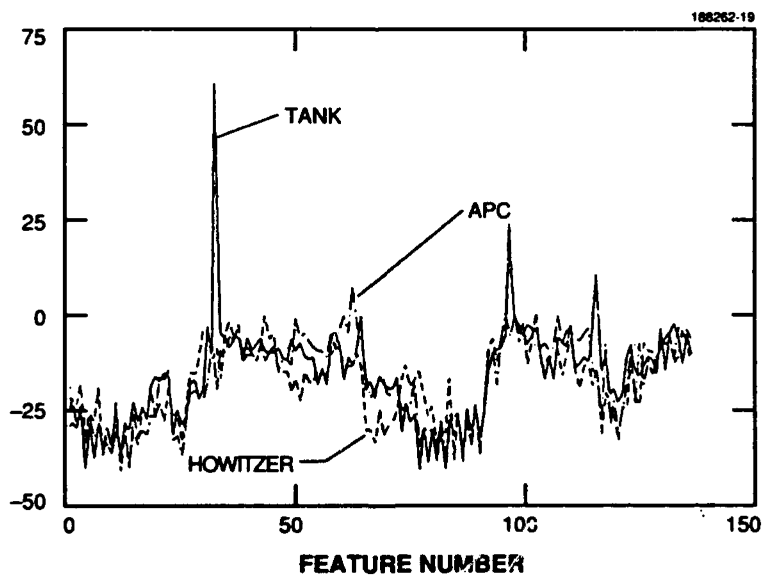


Figure 32. Example feature vectors of a tank, a howitzer, and an armored personnel carrier (APC). The system distinguished similar vectors during training and testing.

The system trained on 10 random tank images that spanned orientations from front-on to end-on and stored the corresponding exemplars in ART-2 nodes 0 through 9. Next the system trained on eight howitzers and stored their exemplars on nodes 10 through 17. During training a Vigilance of 0.999 was used, which corresponded to a 2.6 deg angular separation in feature-vector space.

After the training was completed, the system was tested on the remaining 22 images. Table 4, which summarizes the results, shows that the system correctly classified the remaining 10 tanks and 8 howitzers. When the Vigilance was set to 0.997, which corresponded to an angular separation of 4.4 deg, the four APC images went to an untrained (or unknown) node. Thus, for this example, the system's recognition was errorless.

TABLE 4

Preliminary Classification Results for Military Vehicle

Estimate True	Train	Tank	Howitzer	APC
Tank	10	10	0	0
Howitzer	8	0	8	0
APC	0	0	0	4

6.3.2 Pap Smear Cells

A set of 23 normal and 16 abnormal Pap smear cell images (the Lahey Clinic collaborators judged the cell types) were assembled. The images were 175 x 175 pixels, with 8-bit gray values. Figure 19(a) shows a typical image of a normal cell and Figure 19(b) a typical image of an abnormal cell (the grid suggests the V1 processing). To the right of the photographs, V1 feature values near the cell's nuclei are shown.

To train and test on different orientations, each cell image was rotated 90 deg, 180 deg, and 270 deg. The rotation expanded the dataset to 92 normals and 64 abnormals, or 156 altogether.

Seven typical cells are chosen to set the SUM and V2 multipliers. Figure 33 shows the number of ART-2 categories as the multipliers are varied. The figure shows roughly the boundary. As with the military vehicles, the design values are selected to weight the SUM, V2, and V1 values about equally.

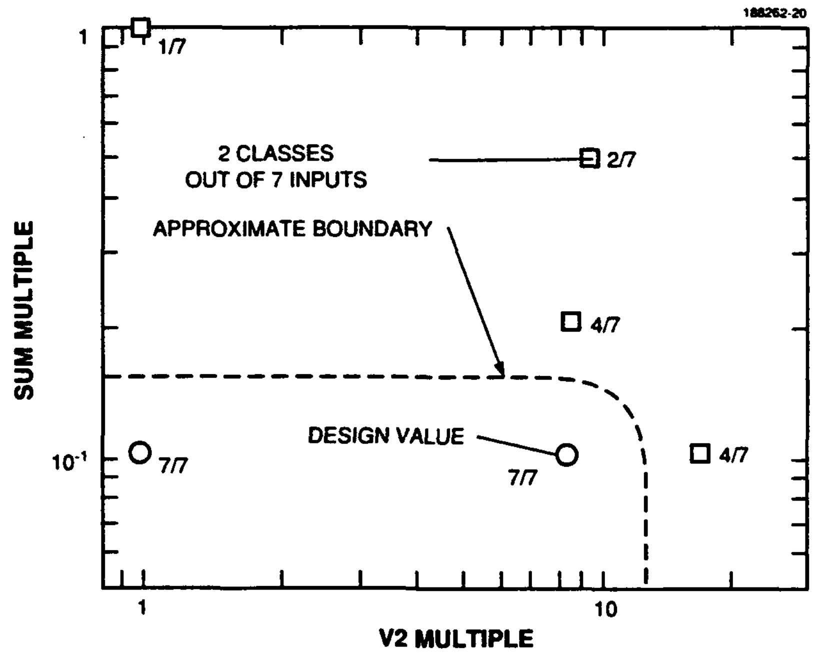


Figure 33. SUM and V2 multiples for Pap smear cervical epithelial cells. As before, at low multiples the number of categories is equal to the number of inputs, but the system is more sensitive to noise. As the multiples are increased the system loses its ability to discriminate, and the number of categories decreases below the number of inputs. The dotted line represents a rough boundary for this transition. A design value inside the boundary and near the elbow of the curve was selected.

The system was trained with a Vigilance of 0.99999. This Vigilance corresponds to 0.256 deg separation in feature space. (Tests showed separation of the cells in feature space varied from 0.256 to about 30 deg.)

At first the training sets were chosen randomly, as had been done with the military-vehicle dataset. The random selection, however, resulted in high error rates, that is, some cells did not make good training examples. In general, cells far from the normal-abnormal boundary in feature space do not help the system improve its discrimination ability. For this reason, an iterative training method was developed that selected cells near the boundary.

The system was trained iteratively by starting with two normal and two abnormal cells. The system was tested on the remaining 152 images. The training set was increased by adding roughly equal numbers of false positives and false negatives. (False positive errors are normal cells classified as abnormal. False negatives are abnormal cells classified as normal.) The error rates of false positives and false negatives drop as the procedure is repeated.

Figure 34 shows the error rates as the number of training examples varies. (Note that it is crucial to keep the false negative rate small to avoid potentially fatal errors.) The curves were produced by the iteration method described above. Each cell took about 40 s to train and about 25 s to test on the Sun/SPARC-CONVEX testbed.

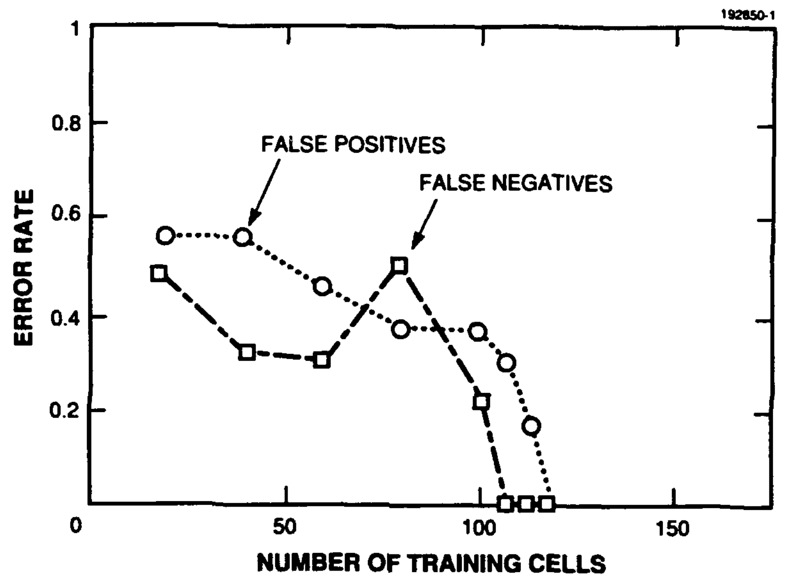


Figure 34. Error rate versus training set size for images of Pap smear cells.

Table 5 summarizes the results and shows no false positives and false negatives with 118 training images.

TABLE 5

Preliminary Cytology Results with Iterative Training

Estimate True	Train	Normal	Abnormal
Normal	66	26	0
Abnormal	52	0	12

The results suggest the system generalizes from its training in the following sense. Mathematically speaking, the feature vectors lie in a 2505-dimensional vector space, as described earlier in Section 4.2.5. The normal cells lie in a subset of that vector space, and the abnormal cells lie in a different subset. If the subsets had formed a checkerboard pattern, or had a highly jagged boundary, training might have required all the images. The curve, however, in Figure 34 suggests the elimination all false negatives with far fewer images. Thus, it is believed that the boundary is comparatively smooth, which allows the system to generalize.

The results suggest the system might have promise for initial cytology screening. Furthermore, the results suggest that the error rates can be deceased to less than, say, five percent, with training sets of several hundred examples for each cell type. More testing is necessary both to confirm or disprove these preliminary results and to assess the system's practical value. For the system to achieve an error rate of less than a few percent, a much larger database is required.

7. APPLICATIONS AND EXTENSIONS

Reliable MV systems have many applications. Besides those areas of interest to MIT Lincoln Laboratory in remote sensing and automatic target recognition, other uses include medical screening, industrial inspection, and robot vision. The architecture of this Lincoln Laboratory system is applicable to these diverse areas.

The basic system architecture also is extendable, and the following sections describe several possibilities. Note that the examples include new principles and so are speculative.

7.1 Sensor Fusion

A direct extension of the research is to combine parallel sensors. Figure 35 shows a fusion concept at the feature-vector level. The bottom system is our basic system with minor additions, and the top is another system, which can be of a different type.

The two systems produce features that train the classifier. Preliminary tests of this concept have been accomplished with video and laser radar range images [21].

7.2 Moving Objects

Another extension is the capability to track and recognize moving objects. Figure 36 shows a conceptual block diagram in which an object in the FOV is moving in an arbitrary direction. To detect the motion, modules can be added that are sensitive to the motion of edges at multiple orientations. These motion detectors mimic characteristics of biological vision systems.

In Figure 36, the system feeds signals from the motion detectors back to the SUPERC module for tracking. The motion-detector features are stored in the feature vector for recognition. To use time-varying features for recognition, the ART-2 module can be replaced by an Avalanche-type NN [22]. This modification would enable the recognition of, say, a flying butterfly [23].

7.3 Binocular Vision

For an extension to binocular vision, Figure 37 shows two of our basic systems working in parallel. In the figure, the SUPERC module points the two "eye balls," and the left and right FOV of each sensor (denoted as L1, L2, R1, and R2, respectively) go separately to two LGNs for calibration and normalization.

The system uses parallel sets of feature detectors, and the feature vector consists of the left and right features and their difference, or disparity. The classifier is similar to the classifier of the baseline system.

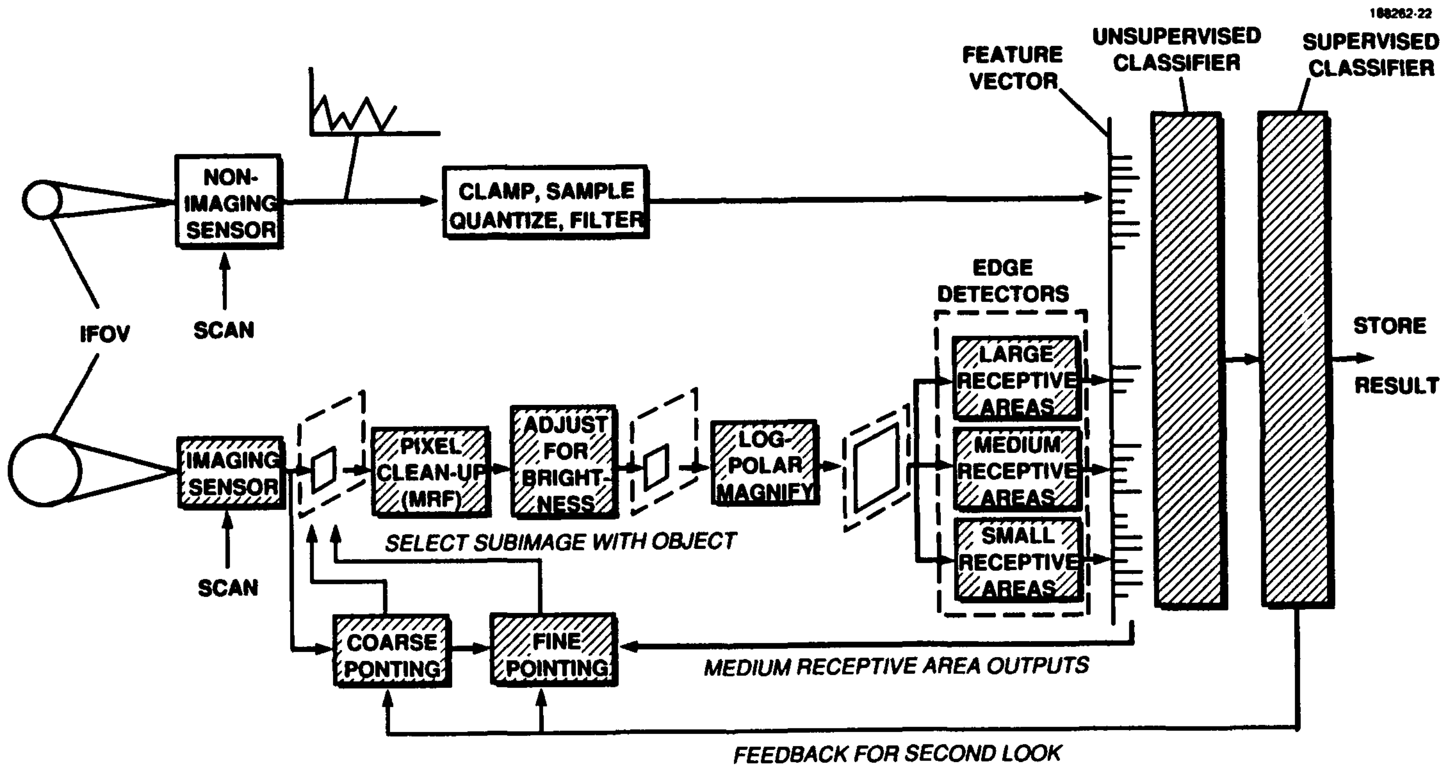


Figure 35. Block diagram showing a system that combines two parallel sensors.

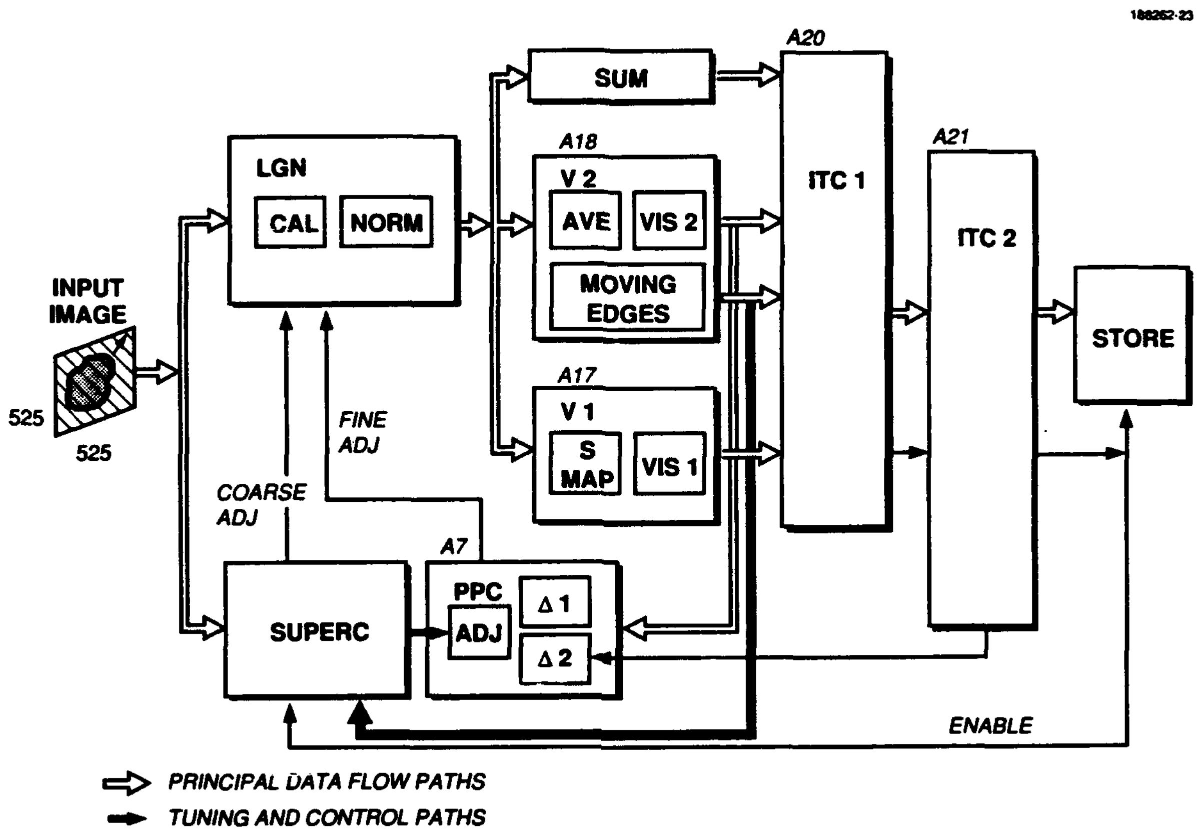


Figure 36. Block diagram of neural network architecture for moving objects.

7.4 Vision-Motor Systems

This section ends on a more speculative note by describing an NN system for driving a car. Figure 38 shows the conceptual block diagram.

The video sensor, with two pointing angles, is part of the basic vision system. (Note that Boston driving needs more than one camera.) An addition to the vision system is the GAZE channel, which gives direction information that combines with the visual features to form the feature vector. The ITC is like that of the baseline system: its output drives a command-generating NN. The network used is Vector Integration to Endpoint (VITE) [24], a type of NN that models biological motor systems. During the learning process, the adaptive elements are in target command map module. Output from the command-generating NN drives other modules that give speed, steering, and braking commands to the automobile.

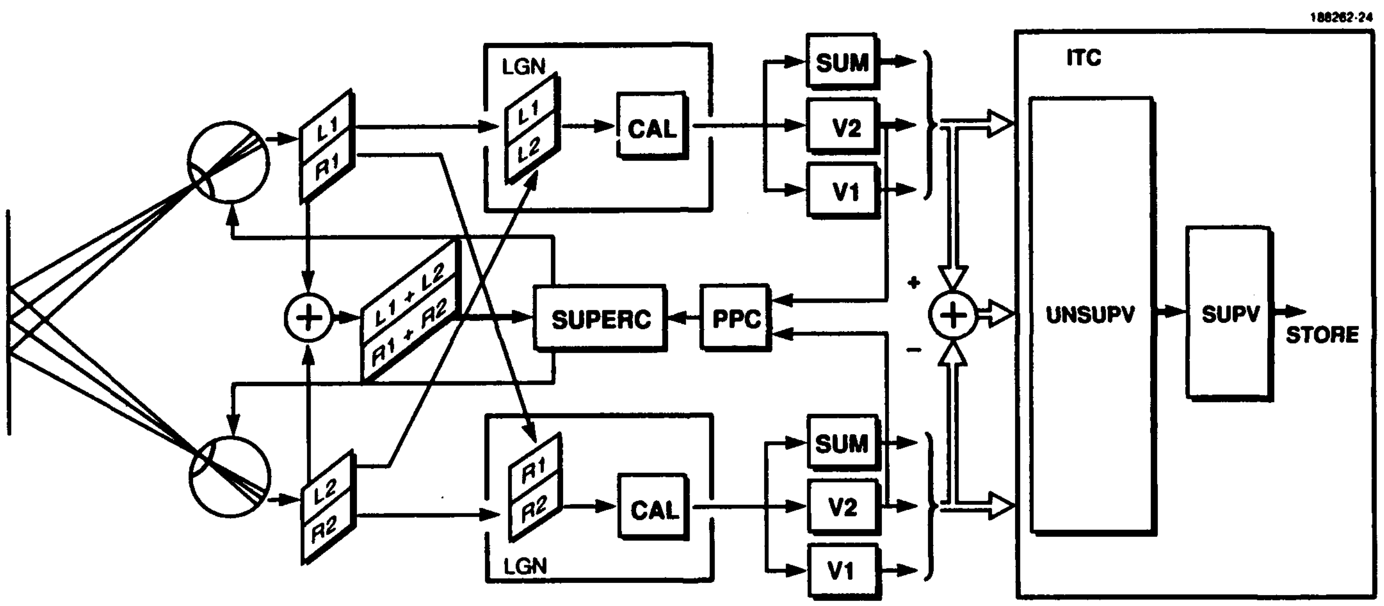


Figure 37. Block diagram of neural network architecture for binocular vision.

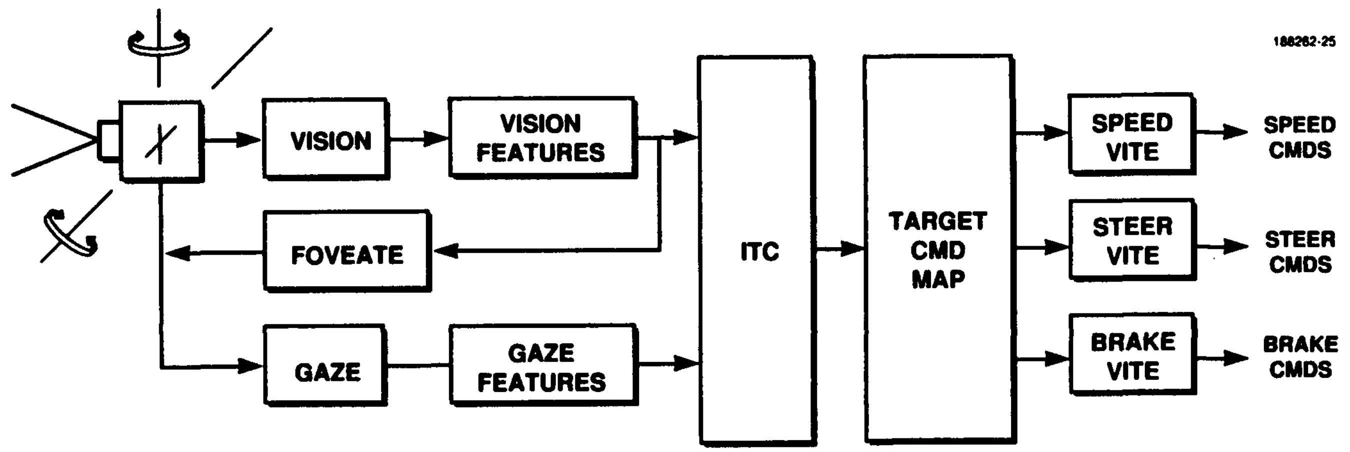


Figure 38. Block diagram of neural network architecture for driving a car. (Note: VITE, or Vector Integration to Endpoint, is a type of neural network that models biological motor systems.)

8. CONCLUSIONS

A general purpose MV system was developed for recognizing stationary visual objects in their natural settings. The system selectively models the architecture and the functions of the human vision system. The recognition of objects is experiential. Moreover, the performance improves through experience.

The system was tested with images of common military vehicles and human cervical cells. The recognition was perfect after sufficient training. This result suggests that further development is warranted and that practical systems might be feasible. Much more work, however, is needed for reducing the design to practice.

Application of the concepts developed in this project have begun. At MIT Lincoln Laboratory, studies continue on additions like motion detection, sensor fusion at the feature level, and applications with imagery from microwave radars and infrared sensors. The work also considers hardware implementations of the algorithm.

This new approach to MV is believed to be very promising.

APPENDIX
DESIGNING NEURAL NETWORKS BY THE GENETIC ALGORITHM

This appendix describes a practical method for designing neural networks with fixed interconnections among the neurons. This class of neural networks is useful for modeling selected biological neural groups and as preprocessors, feature detectors, and control modules in application systems. The approach is from the genetic algorithm, a procedure suggested by natural heredity and evolution for efficiently searching over a parameter space.

The design method avoids common simplifying assumptions, and it enables designing complex, general cooperative-competitive neural networks with feedback that realize (or model) arbitrary, designer-specified, input-output functions. Design examples are given of on-center/off-surround architectures, which crudely model the simple cells in the visual cortex and are applicable as feature detectors in machine vision systems. Scaling laws are described for designing general cooperative-competitive neural networks. The project used this method to design the feature detectors in V1 and V2.

A.1 Introduction

Designing neural networks (NNs) for modeling and applications is an intensive, ongoing activity reported in countless research articles. Many NNs have fixed interconnections among the elementary processors, or neurons. This class of NNs can model selected parts of biological brains (for example, in the visual cortex of higher animals) and are useful in applications (for example, as sensor front-ends, feature detectors, and control units). This appendix shows how complex NNs can be easily designed by a genetic algorithm (GA) method, which have fixed feedforward and feedback connections among the neurons, specified input-output (I/O) characteristics, and, if desired, have properties known from anatomy and physiology.

Conventional NN design techniques make many simplifying assumptions to obtain a tractable problem. Common assumptions include: the neurons are arranged in layers, no lateral connections are made in a layer, and the layer-to-layer signals are feedforward. Or, a neuron can be both excitatory and inhibitory, that is, the connection weights (modeling synaptic transmission coupling) from a neuron to others, can be positive to some and negative to others.

These are significant restrictions, especially in modeling biological systems, because they differ from our present knowledge of the anatomy and physiology of the cerebral cortex of higher animals [8]. The design method described below does not make these assumptions. Thus, it may be of interest to theorists and to designers of applications because the resulting NNs can be smaller for the same function and, perhaps, easier to implement in software or hardware, or both.

A conventional design for NN feature detectors is to compute the output by convoluting the input pattern with a kernel matrix. The kernel matrix is selected so that the output gives, say, a measure of the orientation of a line in the input pattern. The response mimics the orientation responses to illumination contrast of simple cells in the visual cortex [9]. A typical robotics application with this design is described by Kuperstein [25]. While mathematically convenient, the approach is at best a rough approximation of biological feature detectors because convolution is linear.

In feedforward NNs, adjusting connection weights currently is done by simulated annealing (SA) or backpropagation (BP). SA includes the Boltzmann machine [26] and is slow. BP is the most common method. It was originated by Werbos [27], developed, and described by Rumelhart [28] and other members of the PDP group. BP and its many variations, however, suffer from slowness for many problems [29] and is restricted to designing feedforward NNs.

Section A.2 gives a short summary of genetic algorithms, pertinent to the IRP application. Section A.3 derives the representation starting with the additive short-term memory (STM) equation. Section A.4 gives the design method. Section A.5 shows the on-center/off-surround (O/O) design examples and Section A.6 presents scaling laws for general CC NNs.

A.2 Genetic Algorithm Background

For over a decade the GA community, originated by Holland [30], has pursued trial and error strategies for designing adaptive systems. For a collection of current papers see the proceedings of the most recent GA conferences [31,32]. The GA is a search procedure, inspired by evolution and heredity, for finding high performance structures in a complex parameter domain. For NN design purposes, the GA is a search method for finding a good set of weights in a high-dimensional, nonlinear weight space. (A degenerate form of the GA are the well-known gradient-descent techniques, including BP.)

GA theory gives guidelines for constructing practical search techniques (a direct random search of parameter space is not practical because the number of trials increases exponentially with the number of neurons). The fundamental requirements are that the problem be represented by some data structure, the solutions be capable of evaluation, the advances already made be retained, and the population of retained structures be increased [33]. In all GA applications the major problems are: a convenient representation of the system, devising genetic operators that produce good solutions, and defining payoff functions.

The GA can be defined as follows: A set of structures are created (generation), which attempt to solve a problem. The structures (parents) are manipulated by a set of genetic operators (traditionally crossover, inversion, and mutation [30]), to create a new set of structures. The new structures are evaluated on how well they solve the problem. The best set of structures is saved (the next generation). This process is repeated until a structure produces an acceptable solution to the problem.

Researchers have applied the GA to design simple NNs. Recent examples are found in Miller et al. [34], who assume a feedforward NN. A matrix of digits denotes the nature of interconnections among the neurons. The GA picks rows of this matrix and swaps with the parent. The resulting

NN is trained by BP and evaluated. Whitley [35] represents a feedforward NN in binary form with four or eight bits for each connecting weight. The weight bits are concatenated to form a string, which is manipulated by an adaptive mutation operator. The NN is then trained by BP and evaluated. Harper et al. [36] also represents connections by a bit string, uses the standard mutation operator, and trains the system by BP. These GA studies, which only consider simple examples (like the XOR problem), assume feedforward NNs, BP-related performance metrics, and binary string representations for the connection weights.

The initial work with the GA manipulated a representation using bit strings. Application to problems such as NNs requires another representation. Numerous representations have been studied. These include parameter spaces [37], rule spaces of programming languages [38], and structure spaces of classifiers [39]. One example of interest to the NN community is that tree classifiers (modeled by NNs) tend to mutate toward increasing complexity, under certain conditions [39].

Traditionally three genetic operators are used to create the new structures: crossover, inversion, and mutation. The difficulties with these operators are that they require the problem to be represented in powers of two and that some of the mutations will be extremely unlikely [33]. More general operators employing table lookups have been developed.

In any application a task is to devise genetic operators for the problem at hand. The criteria for constructing GA operators is set by the Schema theorem [40]. In general, new operators should not disrupt the distribution of trials and should encourage the formation of building blocks.

Another issue in applying the GA is the *payoff function* in the evaluation stage. GAs are not effective in searching spaces in which the payoff is zero almost everywhere. The design of effective payoff functions is critical, as will be seen later. Vector-values payoff functions may be useful.

Two distinct GA approaches were originated by Holland. These approaches are popularly named after the communities where they were first elaborated.

In the Michigan approach there is a single system consisting of a set of rules or parameters. New rules discovered by genetic operators are applied to existing rules. Each rule is assigned a strength indicating the utility of the rule to the system's goal of obtaining an external payoff. Rules achieve high strength either by obtaining direct payoff from the task environment or by setting the stage for later rules. In the Michigan approach, the GA operates on internal parameters or rules of a single system. The GA picks the "best" as it adapts to an environment.

In the Pittsburgh approach there are numerous structures each with a set of rules. In Holland's original book a particular class of adaptive plans was defined, called reproductive plans [30, pp. 90-111]. In this plan a fixed set of possible solutions is maintained. An individual solution is selected according to its performance rank, modified by one or more genetic algorithms, evaluated by the environment that contains the external inputs, and then used to replace a randomly selected member of the set. In this manner the set of possible solutions evolves to contain members with high performance, because the better an individual performs the more offspring it has.

The Michigan approach has proven to be most practical in on-line, real-time environments, because of the reduced computational loads. The Pittsburgh approach is appropriate with off-line environments, in which more leisurely exploration is acceptable. In GA terminology, this appendix uses a Michigan approach to NN design.

The remainder of the appendix describes an approach for designing general cooperative-competitive (CC) NNs with fixed connection weights. It demonstrates the method by using non-trivial examples (75 neurons, 1920 connection weights) of orientation detectors modeling simple cell modules in the visual cortex of primates, which satisfies important biological constraints.

A.3 Formulation

A.3.1 Activation Equation

Suppose a set of neurons $\{\nu_i\}$ are connected to form a feature detector module. Each neuron is described mathematically by equations that, roughly, model its biological processes. A common approach is to characterize the ν_i-th neuron by its activation level (x_i) and by its connections with other neurons, given by a set of coupling coefficients $\{Z_{ji}\}$.

For the activation level or STM, assume an equation for ν_i of the form [41]:

$$\frac{dx_i}{dt} = -\alpha x_i + \sum_j Z_{ji} f(x_j) + I_i, \tag{A.1}$$

where

x_i = activation of the i-th neuron,

Z_{ji} = LTM trace from the j-th neuron to the i-th neuron,

I_i = external input to the i-th neuron,

$f()$ = a signal function,

α = relaxation time constant parameter.

This equation, called the Additive STM Equation, is basic in NN research and is adequate for many NN designs. (If desired, it can be replaced by the Shunting STM Equation for a better model of the biology [41].)

Assume the coupling coefficients (or the LTM trace) are constant and unknown. For this class of NNs with fixed interconnections, the main problem is to determine a set of weights, $\{Z_{ji}\}$, satisfying prescribed I/O relations.

To illustrate the method, the design of a feature detector module is carried throughout the discussion. Designing NNs for other functions follows similarly.

A simple design of a feature detector module gives a single output for some spatial activation pattern defined on a rectangular array of input neurons. The output could show the angular orientation of the pattern. For a feature detector assume: the input patterns defined on M input neurons, N hidden (internal) neurons, and a single output neuron. Thus, each input pattern is characterized by the activation level of a single output neuron, as shown in Figure A-1.

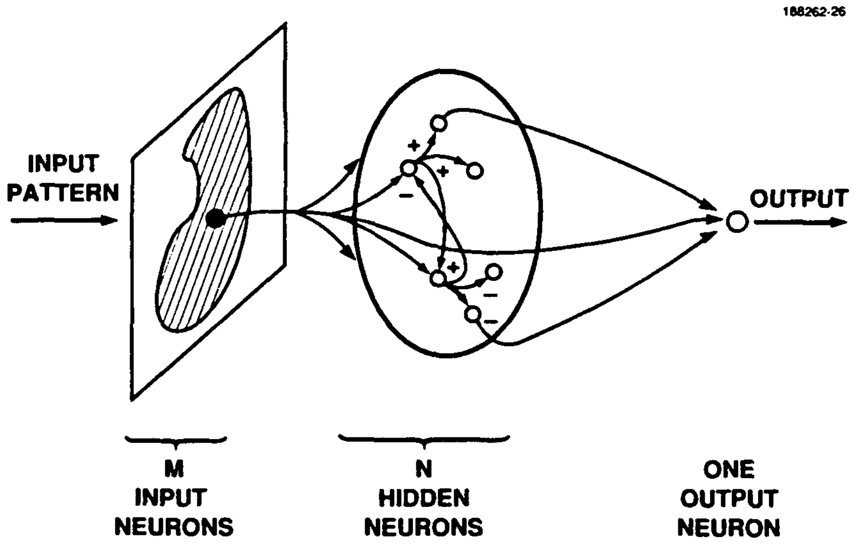


Figure A-1. Nomenclature for a feature detector neural network. An input pattern, shown in cross-hatching, is impressed on M neurons. Each input neuron is connected to N hidden neurons and a single output neuron, whose output is high for a chosen angular orientation of the input pattern and low for other orientations. The hidden neurons may be excitatory (labeled +) or inhibitory (labeled −) and are interconnected. The design method enables computing the connection weights.

The input pattern may be binary or gray. Assume no feedback from the hidden neurons or from the output neuron to the input neurons; however, assume feedback among the hidden neurons. For this example, a set of STM equations can be written from Equation (A.1):

1. <u>Input Neurons:</u>

$$\frac{dx_i}{dt} = -\alpha x_i + I_i,\ i = 1, \cdots, M. \qquad \text{(A.2)}$$

2. Hidden Neurons:

$$\frac{dx_i}{dt} = -\alpha x_i + \sum_{j=1}^{M+N} Z_{ji} f(x_j),\ i = 1, \cdots, N. \tag{A.3}$$

3. Output Neuron:

$$\frac{dx_0}{dt} = -\alpha x_0 + \sum_{j=1}^{M+N} Z_{j0} f(x_j). \tag{A.4}$$

A.3.2 Matrix Formulation

The STM equations of the hidden neurons can be written as:

$$\frac{dx_i}{dt} = -\alpha x_i + \sum_{j=1}^{N} Z_{ji} f(x_j) + \sum_{k=1}^{M} Z'_{ki} f(x_k),\ i = 1, \cdots, N, \tag{A.5}$$

where

Z_{ji} = LTM trace within the hidden neurons,

Z'_{ki} = LTM trace from the input pattern to the hidden neurons.

Assume $\alpha = 1$ (equivalent to rescaling the other variables). Then, in the steady-state the input neuron activations approach the external inputs, that is, $x_k \rightarrow I_k, k = 1, \ldots, M$ as $t \rightarrow \infty$. Assuming no self-sustained oscillations, in the steady-state the hidden neuron activations become:

$$x_i = \sum_{j=1}^{N} Z_{ji} f(x_j) + \sum_{k=1}^{M} Z'_{ki} f(I_k),\ i = 1, \cdots, N. \tag{A.6}$$

For convenience, matrix notation is introduced as follows: Let

$$\mathbf{X} = \begin{bmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix}. \tag{A.7}$$

Here $\mathbf{X}$ is the activation vector of the hidden neurons, written as an $N \times 1$ matrix. The steady-state hidden neuron equations, Equation (A.6), in matrix notation becomes:

$$\mathbf{X} = \mathbf{Af(X)} + \mathbf{Bf(I)} \tag{A.8}$$

where

$$\mathbf{A} = \begin{bmatrix} Z_{11} & \cdots & Z_{N1} \\ . & & . \\ . & & . \\ . & & . \\ Z_{1N} & \cdots & Z_{NN} \end{bmatrix} \tag{A.9}$$

is a constant $N \times N$ matrix,

$$\mathbf{B} = \begin{bmatrix} Z'_{11} & \cdots & Z'_{M1} \\ . & & . \\ . & & . \\ . & & . \\ Z'_{1N} & \cdots & Z'_{MN} \end{bmatrix} \tag{A.10}$$

is a constant $N \times M$ matrix,

$$\mathbf{f(X)} = \begin{bmatrix} f(x_1) \\ . \\ . \\ . \\ f(x_N) \end{bmatrix} \tag{A.11}$$

is an $N \times 1$ matrix with the signal function applied to each element, and

$$\mathbf{f}(\mathbf{I}) = \begin{bmatrix} f(I_1) \\ \cdot \\ \cdot \\ \cdot \\ f(I_M) \end{bmatrix} \tag{A.12}$$

is an $M \times 1$ matrix. Note the indices in $\mathbf{A}$ and $\mathbf{B}$ are reversed from the usual matrix notation because of how the terms were originally defined.

In this formulation no assumptions are made about the kind of signal function. The model also is a "sum-of-sigmoids" and not the usual simplifying approximation of "sigmoid-of-sums."

Let the steady-state output be Z, that is, $Z = x_0(t \to \infty)$. Then, Equation (A.4) gives:

$$Z = \sum_{j=1}^{N} Z_{j0} f(x_j) + \sum_{k=1}^{M} Z'_{k0} f(I_k), \tag{A.13}$$

which in matrix form is:

$$Z = \mathbf{Cf}(\mathbf{X}) + \mathbf{Df}(\mathbf{I}), \tag{A.14}$$

where

$$\mathbf{C} = \begin{bmatrix} Z_{10} & \cdots & Z_{N0} \end{bmatrix} \tag{A.15}$$

is a constant $1 \times N$ matrix, and

$$\mathbf{D} = \begin{bmatrix} Z'_{10} & \cdots & Z'_{N0} \end{bmatrix} \tag{A.16}$$

is a constant $1 \times M$ matrix.

Thus, in matrix notation the NN is described by a set of four matrices $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$.

To design an NN with given I/O characteristics, a set of matrices $\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}\}$ must be determined. With reference to the GA, copies of the system with random changes are simply copies of this matrix set with random changes in their elements. (In the GA context, the matrix elements play a role analogous to the DNA molecules in biological evolution.)

Given a system described by the set $\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}\}$ and an input matrix, $\mathbf{I}$, the hidden system, Equation (A.8), must be solved for the steady-state activation vector, $\mathbf{X}$. Once $\mathbf{X}$ is known, the output Z is computed by Equation (A.14).

A.3.3 Assumptions for Biological-Like Neural Networks

To illustrate the design method, a problem is solved that is difficult by other methods. Assume the NN is to model, crudely, biological feature detectors such as those found in the human primary visual cortex. (Such an NN also is useful in applications because, presumably, the resulting performance would be similar to the comparatively high performances of natural vision systems.) To model biological NNs, assumptions are applied from experimental findings. The assumptions for mimicking biological NNs are discussed by Crick and Asanuma [7] and are summarized here in the first two items. Assume for feature detectors:

1. Each neuron is either type I (excitatory) or type II (inhibitory) and cannot be of both types.
2. A neuron can not excite or inhibit itself.
3. The NN are in a O/O architecture. (The more general CC architecture is considered below.)

With these assumptions, the system matrices for O/O NNs have the following properties:

1. $\mathbf{A}$ has zero diagonal components.
2. $\mathbf{A}$ and $\mathbf{C}$ have negative or zero elements.
3. $\mathbf{B}$ and $\mathbf{D}$ have positive or zero elements.
4. The columns of $\mathbf{A}$ give the lateral inhibition to other hidden neurons (for example, column 1 is the inhibition of neuron ν_1 on the other neurons, and so forth for the other columns).

These constraints are difficult to add in standard design techniques; however, they can be easily accommodated with this approach. Figure A-2 summarizes these constraints on the system matrices.

For the more general CC NN, the hidden neurons are mixtures of excitatory and inhibitory types. Thus, matrices $\mathbf{A}$ and $\mathbf{C}$ have a mixture of columns containing all positive or all negative elements with the diagonal elements of $\mathbf{A}$ still zero.

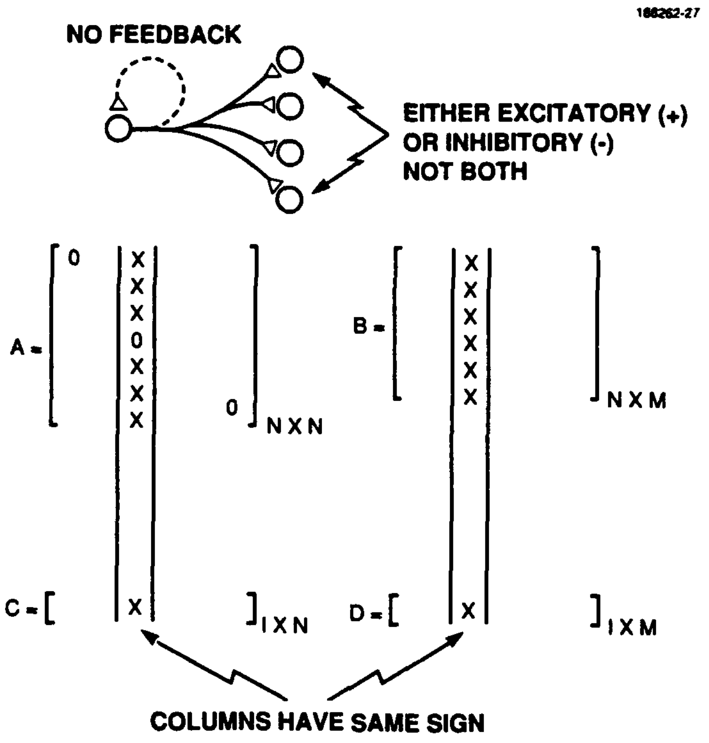


Figure A-2. Constraints of A, B, C, D neural networks. For on-center/off-surround neural networks, the elements in A and C are negative while the elements of B and D are positive. For cooperative-competitive neural networks, the elements in a column and corresponding element in C or D have the same sign, which may be positive or negative.

A.4 Design Procedure

A.4.1 Computational Steps

From the GA, the computational steps for the design procedure are the following:

1. Make copies of the parent set $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$. In each copy, randomly select columns of the parent. The elements of the selected columns are then randomly changed subject to the constraints given in the section A.3.3. In GA terminology, the random changes are done by a mutation operator (see example in Section A.5).
2. Using a set of input training patterns, for each training pattern and each copy, solve for the output. (Note: a solution may not always exist — see Section A.4.3.)
3. Select the best copy according to a payoff criterion (see Section A.4.2). Make this copy the survivor.

4. Using the survivor as the parent for the next generation, repeat steps (1) to (3).
5. Continue until the payoff criterion is met. The surviving system, $\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}\}$, describes the NN.

The procedure is summarized by a flow diagram shown in Figure A-3.

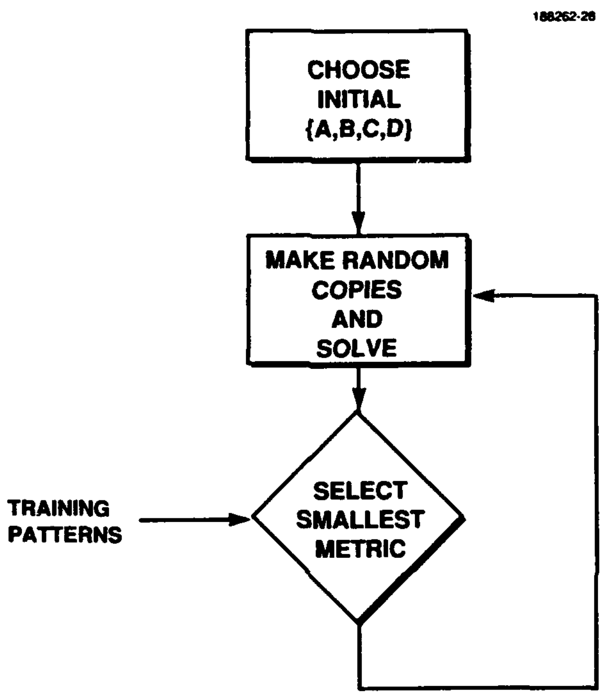


Figure A-3. Computational flow diagram of the design method. The parent set of four matrices $\{A,B,C,D\}$ represents a neural network. At each generation several offspring of the parent are produced by random changes. The best offspring is saved and is the parent for the next generation. The process is continued until a good neural network is found.

A.4.2 Payoff Criterion

Defining a good payoff function is critical, as mentioned in Section A.2. For illustration, a metric is chosen to measure the distance (error) of the output from a desired output, for each training pattern.

Many metrics are possible. One convenient metric specifies that for each input training pattern, the output response lies within a band of upper and lower thresholds, where the thresholds are specified by the designer. This HI-LO metric is formulated as follows:

For N_P training patterns, the metric, d, is:

$$d = \sum_{i=1}^{N_P} (TLO_i \le Z_i \le THI_i), \tag{A.17}$$

where

Z_i = output for the i-th input training pattern,

TLO_i = lower bound for the i-th input pattern,

THI_i = upper bound for the i-th input pattern,

and

$$(TLO_i \le Z_i \le THI_i) = \begin{cases} 0 & \text{if } TLO_i \le Z_i \text{ and } Z_i \le THI_i, \\ 1 & \text{otherwise.} \end{cases} \tag{A.18}$$

Another practical metric specifies that the output be above a passband threshold for certain patterns and below a stopband threshold for the other patterns. This PASS-STOP criterion is formulated below.

For N_1 PASS patterns and N_2 STOP patterns, with $N_P = N_1 + N_2$, the metric, d, is:

$$d = \sum_{i=1}^{N_1} (Z_i \ge T_i) + \sum_{j=1}^{N_2} (Z_j \le T_j), \tag{A.19}$$

where

T_i = selection thresholds, $i = 1, \cdots, N_P$,

and

$$(Z_i \ge T_i) = \begin{cases} 0 & \text{if } Z_i \ge T_i, \\ 1 & \text{otherwise,} \end{cases} \tag{A.20}$$

$$(Z_i \leq T_i) = \begin{cases} 0 & \text{if } Z_i \leq T_i, \\ 1 & \text{otherwise.} \end{cases} \tag{A.21}$$

For the two metrics, the following inequalities hold:

$$0 \leq d \leq N_P. \tag{A.22}$$

The condition $d = N_P$ means the system, $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$, satisfies none of the payoff criteria (maximum error). The condition $d = 0$ means the system satisfies the selection criterion and a solution has been reached. In practice the metric, d, starts at N_P (or smaller) and monotonically decreases to zero.

A.4.3 Solution of the Activation Equation

To implement step (2) of the algorithm (Section A.4.1), for each set $\{\mathbf{A}, \mathbf{B}, \mathbf{I}\}$ solve the equation:

$$\mathbf{X} = \mathbf{A}\mathbf{f}(\mathbf{X}) + \mathbf{B}\mathbf{f}(\mathbf{I}). \tag{A.23}$$

Three outcomes are possible when attempting to solve Equation (A.23): no solutions may exist, a single solution may exist, or multiple solutions may exist. [The three outcomes are easily seen by assuming **I** is binary and solving Equation (A.23) by hand for a low-dimension system.]

Many authors have studied the solution of nonlinear matrix equations such as Equation (A.23) [42]. The method used here is a combination of operator decomposition followed by recursion, typical in fixed-point theorems.

Following Adomian and Adomian [43], the first two terms of an operator decomposition solution, $\mathbf{X_0}$ and $\mathbf{X_1}$, are given by:

$$\mathbf{X_0} = \mathbf{B}\mathbf{f}(\mathbf{I}) \tag{A.24}$$

$$\mathbf{X_1} = \mathbf{A}(\mathbf{X_0}) + \mathbf{X_0} \tag{A.25}$$

The recursion is:

$$\mathbf{X}(+) = \mathbf{A}\mathbf{f}(\mathbf{X}(-)) + \mathbf{B}\mathbf{f}(\mathbf{I}) \tag{A.26}$$

where $\mathbf{X_1}$ is the initial trial solution, and

$\mathbf{X}(-)$ =current trial solution for $\mathbf{X}$,

$\mathbf{X}(+)$ =next trial solution for $\mathbf{X}$.

The recursion is continued until $\mathbf{X}(+) = \mathbf{X}(-)$, or fails after a fixed number of tries.

This recursion converges to a solution if and only if the operator defined by the right-hand side of Equation (23) is a contraction operator [42]. In practice an operator may be a contraction in some subspaces and not in other subspaces. For the examples below, the initial values, $\mathbf{X_1}$, produced by operator-decomposition are in a contraction subspace for about 90 percent of the choices of $\{\mathbf{A},\mathbf{B},\mathbf{I}\}$. It typically requires seven to ten recursions to reach the final solution of the activation equation. [The above scheme for solving Equation (A.23) is preferable to one using only the operator decomposition approach for NNs with many hidden neurons. For small systems with 10 or fewer hidden neurons, the operator decomposition method gives solutions after computing fewer than four terms. See Adomian and Adomian [43] for a description.]

A.5 Examples

Consider designing an O/O NN that is sensitive to horizontal binary patterns, to a resolution of 45 deg, on a rectangular array of neurons. Common design procedures for this problem have simplified topologies, as discussed in Section A.1, or apply other ad hoc approaches.

An example of an ad hoc approach for this problem is to sum the number of ON input neurons in each row of an input rectangular array and pick the maximum. Similarly, sums for the two diagonal and vertical directions are computed for measuring the "strengths" in those directions. A comparison of the four direction values then is a measure of the "orientation" of the pattern. This orientation detector, however, is unsatisfactory because patterns can be easily constructed for which the response is unreasonable.

To illustrate the proposed method, an NN is designed for which a large output shows the input pattern has a horizontal orientation. That is, the weights are chosen so that the NN is sensitive to horizontal edges.

Table A-1 shows the assumed parameters for the example. As seen, the input pattern is defined on 7 × 7 or 49 input neurons. (An NN with an angular resolution of 10 deg would have a larger number of input neurons and could be designed by this method.) The hidden system has 25 neurons.

Starting with a random set $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ for the first generation, at each generation ten copies are made of the parent set. For each copy, a third of the matrix elements (weights) are randomly changed by selecting integer values over the range -10 to $+10$, subject to the constraints given in Section 2.3.

TABLE A-1

Input Parameters fro Designing a Horizontal Feature Detector Using and On-Center/Off-Surround Neural Network

Example 1	Parameters
Input Neurons (*M*)	7 × 7 (49)
Hidden Neurons (*N*)	25
Copies per Generation	10
Fraction of Wwights Changed	
Per Copy	1/3
Search Range	0, ±1,±2,..., ±10
High Band	50 to 100
Low Band	−100 to 10

A HI-LO metric is assumed for the training patterns with some of the outputs in a high band and the others in a low band. The desired output response to the high (horizontal) training patterns is 50 to 100. The desired response to the low (nonhorizontal) training patterns is -100 to 10. The input vector, **I**, for each training pattern is produced by row-by-row scanning. For this NN, 1924 coefficients must be specified. For simplicity, the signal function is a unit step.

The HI-LO metrics cf the ten copies are computed and compared with the metric of the parent set. If an offspring metric is below the parent metric, the offspring replaces the parent set for the next generation.

Figure A-4 shows the assumed 12 training patterns and the desired responses (high or low) for each pattern. As seen, training is on three horizontal patterns and nine other patterns, and each training pattern has about two hidden neurons.

The design algorithm was coded in the APL*PLUS programming language and run on an 8 MHz IBM PC/AT machine. Figure A-6 shows the history of the metric as the system evolves

188262-29

$$\boxed{\begin{matrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\1&1&1&1&1&1&1\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{matrix}} \qquad \boxed{\begin{matrix}0&0&0&0&0&0&1\\0&0&0&0&0&1&0\\0&0&0&0&1&0&0\\0&0&0&1&0&0&0\\0&0&1&0&0&0&0\\0&1&0&0&0&0&0\\1&0&0&0&0&0&0\end{matrix}}$$

$$\boxed{\begin{matrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&1&1&1&1&1&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{matrix}} \qquad \boxed{\begin{matrix}0&0&0&0&0&0&0\\0&0&0&0&0&1&0\\0&0&0&0&1&0&0\\0&0&0&1&0&0&0\\0&0&1&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&0&0&0\end{matrix}}$$

$$\boxed{\begin{matrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&1&1&1&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{matrix}} \qquad \boxed{\begin{matrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&1&0&0\\0&0&0&1&0&0&0\\0&0&1&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{matrix}}$$

HORIZONTAL **45 DEG**

$$\boxed{\begin{matrix}0&0&0&1&0&0&0\\0&0&0&1&0&0&0\\0&0&0&1&0&0&0\\0&0&0&1&0&0&0\\0&0&0&1&0&0&0\\0&0&0&1&0&0&0\\0&0&0&1&0&0&0\end{matrix}} \qquad \boxed{\begin{matrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&1\end{matrix}}$$

$$\boxed{\begin{matrix}0&0&0&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&1&0&0&0\\0&0&0&1&0&0&0\\0&0&0&1&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&0&0&0\end{matrix}} \qquad \boxed{\begin{matrix}0&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&0\end{matrix}}$$

$$\boxed{\begin{matrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&1&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{matrix}} \qquad \boxed{\begin{matrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{matrix}}$$

VERTICAL **135 DEG**

Figure A-4. Example binary training patterns for a feature detector. Three patterns are used for each of four orientations. For sensitivity to horizontal patterns, the outputs are specified high for the three horizontal patterns and low for the others.

to a solution in about 600 generations. The metric started at $d = 10$ and ended at $d = 1$ at 600 generations. The one remaining error was a response above the high threshold, and so the run was stopped.

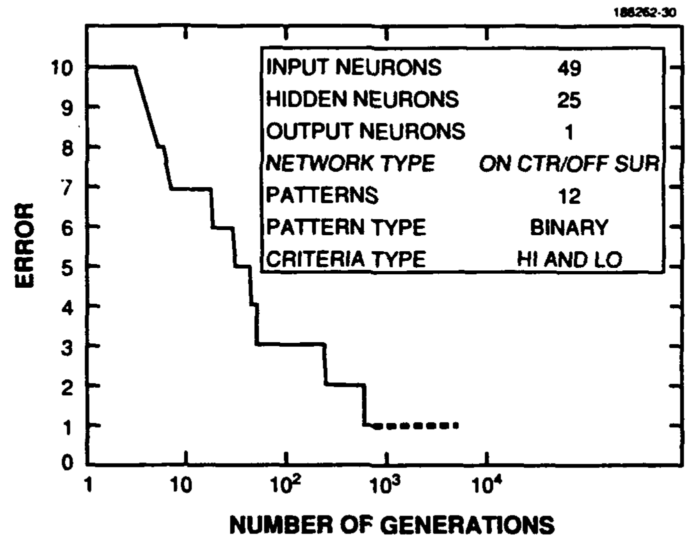


Figure A-5. Time history of the error that measures the deviation of the outputs of 12 training patterns from specified outputs. The box shows the assumptions for a horizontal detector with an on-center/off-center surround architecture.

Figure A-6 shows the resulting system's response to the training patterns. It shows the ratio of high-to-low values (the effective "signal-to-noise ratio") is above five.

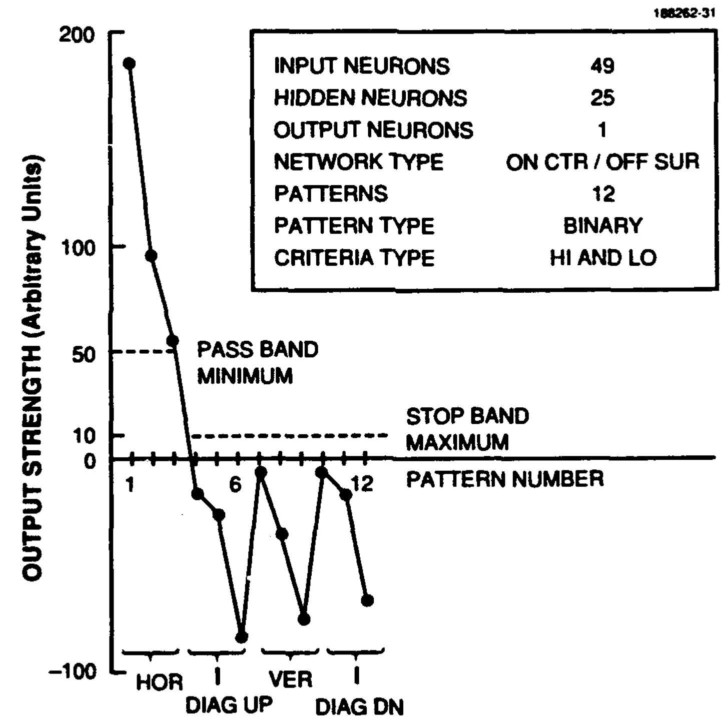
The example was repeated with a different payoff criterion to design a diagonal-sensitive (45 deg) NN. All parameters are the same as in the horizontal example. The high responses are the 45 -deg training patterns and the others have a low response. Figure A-7 shows the responses of a diagonal feature detector to the training patterns. Thus, the method easily produced designs for orientation detectors (with 45-deg angular resolution) satisfying biological constraints.

The design technique also can be applied to gray input images by a straightforward extension to the preceding algorithm. The input values are scaled to lie in the interval [0,1] by preprocessing. To preserve the grayness, a piecewise-linear signal function with saturation is used in the input terms to give:

$$\mathbf{X} = \mathbf{A}\mathbf{f}_1(\mathbf{X}) + \mathbf{B}\mathbf{f}_2(\mathbf{I}) \tag{A.27}$$

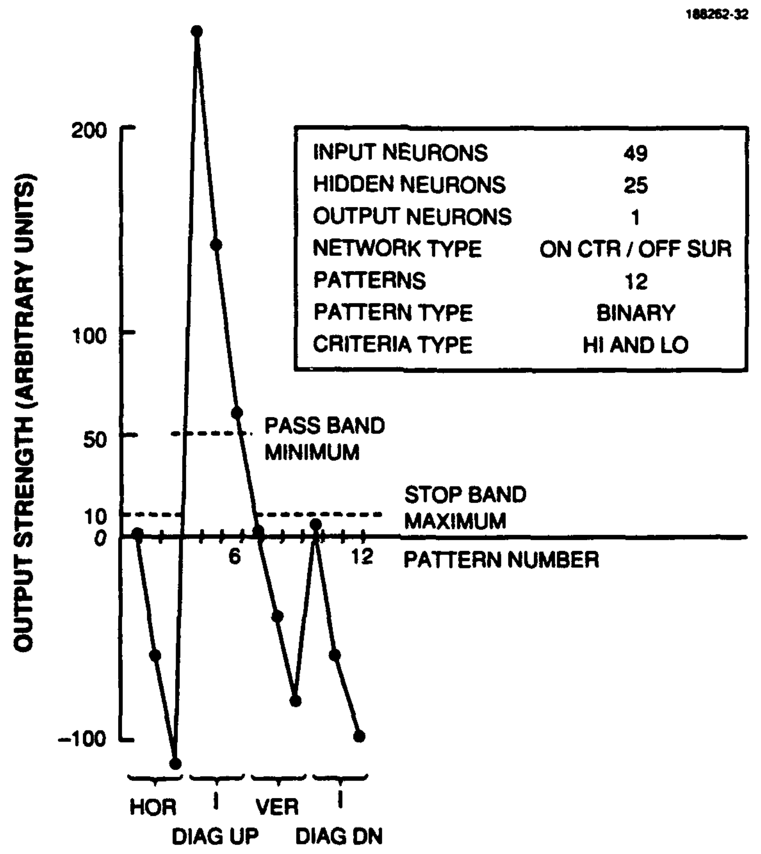
$$Z = \mathbf{C}\mathbf{f}_1(\mathbf{X}) + \mathbf{D}\mathbf{f}_2(\mathbf{I}) \tag{A.28}$$

where $\mathbf{f}_1()$ may be a unit step like before and $\mathbf{f}_2()$ is a piecewise-linear signal with saturation such as



INPUT NEURONS	49
HIDDEN NEURONS	25
OUTPUT NEURONS	1
NETWORK TYPE	ON CTR / OFF SUR
PATTERNS	12
PATTERN TYPE	BINARY
CRITERIA TYPE	HI AND LO

Figure A-6. Response of a horizontal feature detector to 12 training patterns. The three horizontal patterns produced outputs above 50 and the others produced outputs below 10.



INPUT NEURONS	49
HIDDEN NEURONS	25
OUTPUT NEURONS	1
NETWORK TYPE	ON CTR / OFF SUR
PATTERNS	12
PATTERN TYPE	BINARY
CRITERIA TYPE	HI AND LO

Figure A-7. Response of a 45-deg diagonal feature detector to 12 training patterns. The same parameters and payoff function were used as for the horizontal detector except the high outputs were the 45-deg patterns and the low outputs were the others.

$$\mathbf{f_2(X)} = \begin{cases} \mathbf{0}, & \mathbf{X < 0} \\ \mathbf{X}, & \mathbf{0 \le X \le 1} \\ \mathbf{1}, & \mathbf{X > 1.} \end{cases} \tag{A.29}$$

This set of equations then is used in the design procedure like before.

The parameters in Table A-1 must be consistent or no solution is possible. In GA terminology, the payoff function must be capable of being met by a search over the parameter space. The examples and the scaling laws of the next section give rough values. In practice the parameters can be readily selected during initial trials.

A.6 Scaling Laws for Cooperative-Competitive Neural Networks

In practice minimizing computation time is desirable. To produce rough guidelines, many design exercises of orientation detectors were run for fixed CC NN, that is, the hidden neurons may be of both types I and II (excitatory and inhibitory, respectively). (In comparison, the O/O NNs in the preceding section have only type II hidden neurons.) The CC NNs are more general, and as a result they can satisfy I/O requirements that O/O NNs can not.

In a CC NN, the system matrix properties for O/O NN are changed as follows (see Section A.3.3):

1. Property (1) still holds.
2. Properties (2) and (3) are changed so that the corresponding columns of **A** and **C** and **C** and **D** have the same sign.

Two important parameters in the algorithm are the number of hidden neurons and the number of copies made per generation. A relationship is written assuming a power law:

$$NG \propto N^{\alpha} NC^{\beta} \tag{A.30}$$

where

NG = number of generations,

N = number of hidden generations,

NC = number of copies per generation.

Figures A-8 and A-9 show the number of generations required to find a solution as a function of the number of hidden neurons and the copies made per generation, respectively. The brackets show the range of the number of generations needed to reach a solution. As shown for the CC NNs, $\alpha \approx 1$ and $\beta \approx -3/2$.

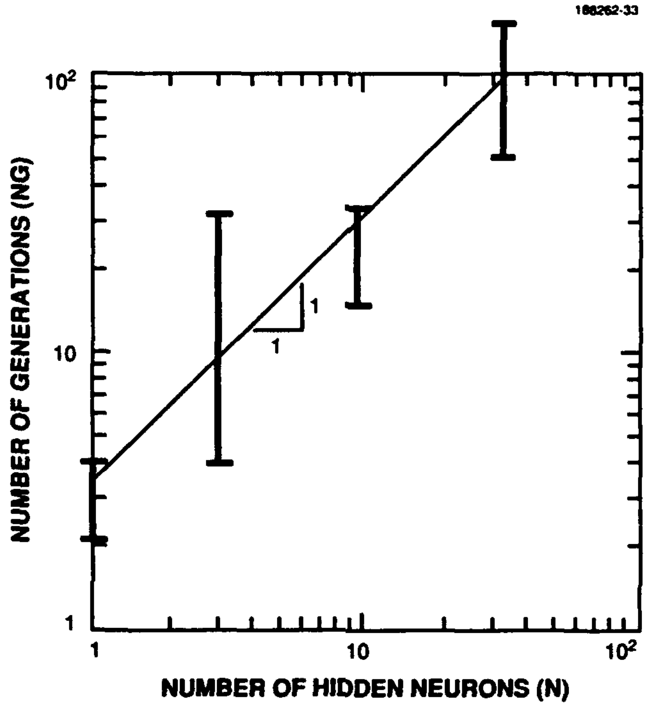


Figure A-8. Scaling law for cooperative-competitive neural networks. The vertical axis shows the number of generations needed to satisfy the payoff function as the number of hidden neurons was varied. The brackets indicate the range in values, which is a random variable.

To first approximation the computational time varies like $N/NC^{1/2}$, because the total computer time is proportional to the total number of copies, that is, to $NG \times NC$. In practice, the number of generations needed for solution is random, because the mutation process is random, and the guidelines show the mean number of generations.

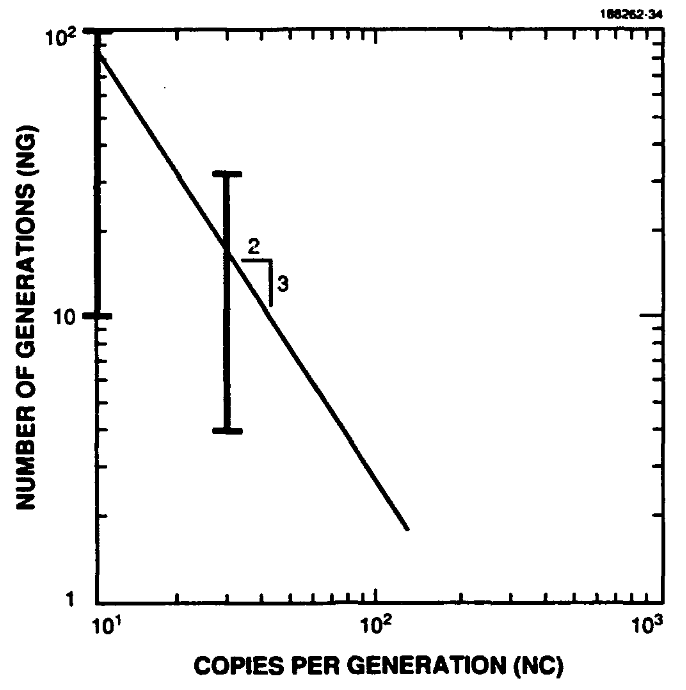


Figure A-9. Scaling law for cooperative-competitive neural networks. The vertical axis shows the number of generations needed to satisfy the payoff function as the number of copies per generation was varied. The brackets indicate the range in values, which is a random variable.

A.7 Discussion

This appendix gives a practical, unified method for designing complex, fixed interconnected NNs that realizes designer-specified I/O characteristics, and (if desired) meets constraints emerging from the experimental studies of natural brains. The previous sections describe the design of O/O and CC NNs meeting specified I/O functions according to two criteria. Many examples show the approach produces NN designs with good output signal-to-noise ratios.

The method has several extensions of interest to theorists and application designers. In the applications, the technique can be applied to design multiple output NNs with specified I/O properties. Preliminary designs have been done of NNs with two outputs for a control system. The results will be reported in the future. For the theorists, the modeling of veto cells and neural systems with diffuse inputs, described in J.P. Frisby [8], can be carried out.

REFERENCES

1. I.E. Gordon, "Marr's Computational approach to visual preception" in *Theories of Visual Perception*, New York, NY: John Wiley & Sons (1989), Chap. 8.
2. A. Rosenfeld, "Image analysis: problems, progress and prospects," in M.A. Fischler and O. Firschein (eds.), *Readings in Computer Vision*, Los Altos, CA: Morgan Kaufmann Publishers Inc. (1987), pp. 3-12.
3. A. Rosenfeld, "Computer vision: basic principles," in *IEEE Proceedings, 76 (8)*, 863-869 (1988)
4. D. Marr, *Vision*, San Francisco, CA: W.H. Freeman and Company (1982).
5. H. Freeman (ed.), *Machine Vision-Algorithms, Architectures, and Systems*, Boston, MA: Academic Press (1988).
6. C.H. Bailey and P. Gouras, "The retina and phototransduction," in E.R. Kandel and J.H. Schwartz (eds.), *Principles of Neural Science*, 2nd edition, New York: Elsevier North-Holland (1981), pp. 344-356.
7. F.H.C. Crick and C. Asanuma, "Certain aspects of the anatomy and physiology of the cerebral cortex," in J.L. McCelland and D.E. Rumelhart (eds.), *Parallel Distributed Processing*, Cambridge, MA: The MIT Press (1988), vol. 2, pp. 333-371.
8. J.P. Frisby, *Seeing-Illusion, Brain and Mind*, New York, NY: Oxford University Press (1980).
9. D.H. Hubel, *Eye, Brain, and Vision*, New York, NY: W.H. Freeman and Company (1988).
10. E.R. Kandel, "Processing of form and movement in the visual system, " in E.R. Kandel and J.H. Schwartz (eds.), *Principles of Neural Science*, 2nd edition, New York, NY: Elsevier North-Holland (1981), pp. 366-383.
11. J.P. Kelly, "Anatomy of the central visual pathways," in E.R. Kandel and J.H. Schwartz (eds.), *Principles of Neural Science*, 2nd edition, New York, NY: Elsevier North-Holland (1981), pp. 356-365.
12. S.W. Kuffler, J.G. Nicholls, and A.R. Martin, *From Neuron to Brain*, 2nd edition, Sunderland, MA: Sinauer Associates Inc. Publishers (1984).
13. J.H. Maunsell, "Physiological evidence for two visual subsystems," in L. Vaina (ed.), *Matters of Intelligence*, Dordrecht, Holland: D. Reidel Publishing Co. (1987), pp. 59-88.
14. T.B. Sheridan and W.R. Ferrell, *Man-Machine Systems: Information, Control and Decision Models of Human Performance*, Cambridge, MA: The MIT Press (1974) Ch. 13.
15. D.C. Van Essen and J.H.R. Maunsell, " Hierarchical organization and functional streams in the visual cortex," *Trends Neurosci 6* (9), 370-375, (1983).

16. E.W. Kent, *The Brains of Men and Machines*, New York, NY: McGraw-Hill Publications Co. (1981).

17. G.A. Carpenter and S. Grossberg, "ART 2: Self-organization of stable category recognition codes for analog input patterns," *Applied Optics, 26 (12)*, 4919-4930 (1987).

18. M.M. Menon, private communication.

19. P.J. Kolodzy and J.E. Baum, " Logical implementation of the automatic target recognition working group (ATRWG) 9-track tape format image storage format," MIT Lincoln Laboratory, Lexington, Mass., Technical Rep. 920, (1989). DTIC AD-A236627.

20. R. Walton, private communication.

21. R.L. Harvey and K.G. Heinemann, "A biological vision model for sensor fusion," *4th National Symposium on Sensor Fusion*, American Institute of Aeronautics and Astronautics, Orlando FL (1991).

22. S. Grossberg, *Studies of Mind and Brain*, Boston, MA: D. Reidel Publishing Co. (1982).

23. K.A. Martin and V.H. Perry, "On seeing a butterfly: The physiology of vision," *Sci. Prog. Oxf., 72*, 259-280 (1988).

24. D. Bullock and S. Grossberg, "Neural dynamics of planned arm movements: Emergent invariants and speed-accuracy properties during trajectory formation," in S. Grossberg (ed.), *Neural Networks and Natural Intelligence*, Cambridge, MA: The MIT Press (1988), pp. 553-622.

25. M. Kuperstein, "Neural model of adaptive hand-eye coordination for single postures," *Science 239*, 1308-1311 (1988).

26. D. Ackley, G. Hinton, and T. Sejowski, "A learning algorithm for Boltzmann machines,"*Cognitive Science 9*, 147-169 (1985).

27. P.J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Doctoral Dissertation, Applied Mathematics, Harvard University (1974).

28. D. Rumelhart and J. McClelland, *Parallel Distributed Processing*, Cambridge, MA: MIT Press (1986), vol. 1 (pp. 318-364).

29. R. Hecht-Nielsen, *Neurocomputing*, New York, NY: Addison-Wesley Publishing Company, Inc. (1990), p. 137.

30. J. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: The University of Michigan Press (1975).

31. J. Grefenstette,"Genetic algorithms and their applications," in *Proceedings Second International Conference on Genetic Algorithms and Their Applications*, Cambridge, MA (1987).

REFERENCES
(Continued)

32. J. Schaffer (ed.), "Genetic algorithms and their applications," in *Proceedings Third International Conference on Genetic Algorithms and Their Applications*, San Mateo, CA: Morgan Kaufmann Publishing Co. (1989).

33. A. Bickel and R. Bickel, "Tree structure rules in genetic algorithms," in *Proceedings Second International Conference on Genetic Algorithms and Their Applications*, Cambridge, MA (1987), pp. 77-81.

34. G.F. Miller, P.M. Todd, and S.U. Hegde, "Designing neural networks using genetic algorithms," in *Proceedings of the Third International Conference on Genetic Algorithms*, J.D. Schaffer (ed.), San Mateo, CA: Morgan Kaufmann Publishers, Inc. (1989), pp. 379-384.

35. D. Whitley and T. Hanson, "Optimizing neural networks using faster, more accurate genetic search," in *Proceedings of the Third International Conference on Genetic Algorithms*, J.D. Schaffer (ed.), San Mateo, CA: Morgan Kaufmann Publishers, Inc. (1989), pp. 391-396.

36. S.A. Harp, T. Samard, and A. Guha, "Toward the genetic synthe^s^is of neural networks," in *Proceedings of the Third International Conference on Genetic Algorithms*, J.D. Schaffer (ed.), San Mateo, CA: Morgan Kaufmann Publishers, Inc. (1989), pp. 360-369.

37. L. Davis and S. Coombs, "Genetic algorithms and communication link speed design," in *Proceedings Second International Conference on Genetic Algorithms and Their Applications*, Cambridge, MA (1987), pp. 252-260.

38. J. Schaffer and A. Morishima, "An adaptive crossover distribution mechanism for genetic algorithms," in *Proceedings Second International Conference on Genetic Algorithms and Their Applications*, Cambridge, MA (1987), pp. 36-49.

39. C. Dolan, and M. Dyer, "Towards the evolution of symbols," in *Proceedings Second International Conference on Genetic Algorithms and Their Applications*, Cambridge, MA (1987), pp. 123-131.

40. K. De Jong, "On using genetic algorithms to search program spaces,"in *Proceedings Second International Conference on Genetic Algorithms and Their Applications*, Cambridge, MA (1987), pp. 210-216.

41. S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architectures," *Neural Networks 1*(1), 17-61 (1988).

42. K. Deimling, *Nonlinear Functional Analysis*, New York, NY: Springer-Verlag (1985), pp. 186-216.

43. G. Adomian and G.E. Adomian,"A global method for solution of complex systems,"*Mathematical Modeling 5*, 251-263 (1984).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE 21 July 1992	3. REPORT TYPE AND DATES COVERED Technical Report

4. TITLE AND SUBTITLE

Neural Network Architectures for Growth Image Recognition

5. FUNDING NUMBERS

C — F19628-90-C-0002
PR — 318

6. AUTHOR(S)

Robert L. Harvey, Paul N. DiCaprio, and Karl G. Heinemann

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Lincoln Laboratory, MIT
P.O. Box 73
Lexington, MA 02173-9108

8. PERFORMING ORGANIZATION REPORT NUMBER

TR-955

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

ESC-TR-92-100

11. SUPPLEMENTARY NOTES

None

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

As part of Lincoln Laboratory's research on neural network technology, a general-purpose machine vision system was designed that can learn to recognize diverse objects. This system models human vision, primarily with neural networks, and the learning is by example. The system was tested on two disparate classes of objects, military vehicles and human cells, with video images of natural scenes. These objects were chosen because large databases are available and because most researchers judge them to be unrelated. The system was trained and tested with 40 images of military vehicles. After training with 18 images, the system was able to recognize the tanks, howitzers, and armored personnel carriers of the remaining images without error. Pathologists at Lahey Clinic Medical Center collaborated in the cytology study where the system was trained and tested on 156 cell images from human cervical Pap smears. After training with 118 images, the system correctly classified all of the other cells as normal or abnormal (that is, precancer). These results are preliminary because the number of military vehicles and Pap smear samples was small. Nonetheless, the results are extremely encouraging and clearly indicate that additional development of the system is warranted. We note that the architecture of the system is applicable to many civilian and military tasks. The application of the system to a specific task requires appropriate training.

14. SUBJECT TERMS

classification	pap smears
neural net	genetic algorithms
pattern classification	machine vision

15. NUMBER OF PAGES

122

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by AMSI Std. 239-18
298-102